NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A PROTOTYPE EXPERT SYSTEM TO FORECAST
TYPHOON CONDITIONS
AT CUBI POINT, PHILIPPINES

by

Bruce M. Hagaman

September 1988

Thesis Advisor R. L. Elsberry

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings	
Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.	
Declassification Downgrading Schedule			
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
Name of Performing Organization aval Postgraduate School	6b Office Symbol *(if applicable)* 63	7a Name of Monitoring Organization Naval Postgraduate School	
Address *(city, state, and ZIP code)* onterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No / Project No / Task No / Work Unit Accession No	

Title *(include security classification)* A PROTOTYPE EXPERT SYSTEM TO FORECAST TYPHOON CONDITIONS AT UBI POINT, PHILIPPINES

Personal Author(s) Bruce M. Hagaman

Type of Report aster's Thesis	13b Time Covered From To	14 Date of Report *(year, month, day)* September 1988	15 Page Count 91

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or po-ion of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
ld	Group	Subgroup	tropical cyclone, tropical cyclone forecasting

Abstract *(continue on reverse if necessary and identify by block number)*

A prototype expert system is designed to forecast the tropical cyclone related winds that may be used to set Conditions Readiness (COR) at Cubi Point, Philippines. One set of rules modifies the storm position and strength forecasts to account terrain interactions while crossing the Philippines. A second set estimates the local winds given the modified storm posi-n and intensity.

Tests using an independent storm set indicate the terrain-modified positions are comparable in accuracy to current Joint phoon Warning Center forecasts. However, the reduction of storm intensity due to terrain is underestimated and the stward translation of the storm is reduced too much. Finally, the conservative strategy of using worst-case wind gust es-nates also contributes to an overprediction of the local winds and thus the COR. COR estimates are only 32% accurate, th a 95% capture rate for winds over 35 kt but a 70% false alarm rate due to overforecast of winds under 35 kt. More namic/statistical study appears to be required to refine the terrain-modification algorithm. Empirical rules from expert ecasters should be included in future systems.

Distribution/Availability of Abstract unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
Name of Responsible Individual L. Elsberry	22b Telephone *(include Area code)* (408) 646-2373	22c Office Symbol 63Es

FORM 1473,84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

A Prototype Expert System to Forecast Typhoon Conditions at Cubi Point, Philippines

by

Bruce M. Hagaman
Lieutenant, United States Navy
B.S., United States Naval Academy, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN METEOROLOGY AND OCEANOGRAPHY

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

ABSTRACT

A prototype expert system is designed to forecast the tropical cyclone related winds that may be used to set Conditions of Readiness (COR) at Cubi Point, Philippines. One set of rules modifies the storm position and strength forecasts to account for terrain interactions while crossing the Philippines. A second set estimates the local winds given the modified storm position and intensity.

Tests using an independent storm set indicate the terrain-modified positions are comparable in accuracy to current Joint Typhoon Warning Center forecasts. However, the reduction of storm intensity due to terrain is underestimated and the westward translation of the storm is reduced too much. Finally, the conservative strategy of using worst-case wind gust estimates also contributes to an overprediction of the local winds and thus the COR. COR estimates are only 32% accurate, with a 95% capture rate for winds over 35 kt but a 70% false alarm rate due to overforecast of winds under 35 kt. More dynamic/statistical study appears to be required to refine the terrain-modification algorithm. Empirical rules from expert forecasters should be included in future systems.

TABLE OF CONTENTS

I. INTRODUCTION 1
A. CURRENT NEEDS 1
B. THE NAVY TYPHOON PREDICTION PROBLEM 2
C. THE EXPERT SYSTEM APPROACH 4
D. PREDESIGN DECISIONS 5

II. DEFINING THE PROBLEM 7
A. PROTOTYPE SCOPE 7
B. SELECTING THE GOAL 8

III. GENERATING THE RULE BASE 11
A. INTRODUCTION 11
B. DETERMINING LOCAL CUBI POINT WINDS 11
C. INTENSITY MODIFICATION 13
D. POSITION MODIFICATION 15
E. CLIMATOLOGICAL STORM TRACKS 21

IV. THE PROLOG ALGORITHM 24
A. UNMODIFIED FORECASTS 24
B. MODIFYING FORECASTS 25
C. CLIMATOLOGICAL TRACKS 27
D. EXITING THE PROGRAM 28

V. TESTING THE EXPERT SYSTEM 32
A. DATA SET 32
B. GENERAL TEST PROCEDURES 34
C. WIND SPEED ESTIMATES 34
1. Test Procedure 34
2. Wind Speed Results 35
D. INTENSITY AND SPEED MODIFICATIONS 39
1. Test Procedure 39

2. Intensity and Translation Speed Modification Results 40
E. POSITION MODIFICATION 46
1. Test Procedure 46
2. Position Errors 46
3. COR Predictions 47
F. CLIMATOLOGICAL TRACKS 49

VI. SUMMARY AND CONCLUSIONS 51
A. SUMMARY OF RESULTS 51
B. AREAS NEEDING IMPROVEMENT 53
C. CONCEPT VALIDATION 54

APPENDIX A. TURBO PROLOG PRIMER 56
A. INTRODUCTION 56
B. PROLOG FACTS 56
C. PROLOG RULES AND BACKTRACKING 58

APPENDIX B. TROPICAL CYCLONE CONDITION FORECASTING PROGRAM 62

REFERENCES 79

INITIAL DISTRIBUTION LIST 81

LIST OF TABLES

Table 1. CUBI POINT CONDITION OF READINESS CRITERIA 10
Table 2. STORMS USED TO TEST ALGORITHM 34
Table 3. WIND PREDICTION ERROR STATISTICS 36
Table 4. PREDICTED VS. APPROPRIATE COR SETTINGS 49
Table 5. ERROR MAGNITUDE SUMMARY (TY AND TS) 51

LIST OF FIGURES

Fig. 1. General program structure 12
Fig. 2. Segments used in maximum and mean gust calculations 14
Fig. 3. Average intensity profiles 15
Fig. 4. Average intensity profiles 16
Fig. 5. Average translation speed profiles 18
Fig. 6. Average translation speed profiles 19
Fig. 7. Cross-track forecast errors 20
Fig. 8. Climatological storm tracks 22
Fig. 9. Program main option menu 25
Fig. 10. Secondary menu (after choosing Option 1) 26
Fig. 11. Secondary menu (after choosing Option 2) 28
Fig. 12. Illustration of the Option 2 Option 4 procedure 29
Fig. 13. Illustration of the Option 2 Option 5 procedure 30
Fig. 14. Post-run summary display screen 31
Fig. 15. Relationship between wind types 33
Fig. 16. Scatterplot indicating wind forecast errors 38
Fig. 17. Intensity forecast errors (TY and TS) 42
Fig. 18. Intensity forecast errors (TY only) 43
Fig. 19. Speed error statistics (TY and TS) 44
Fig. 20. Speed error statistics (TY only) 45
Fig. 21. Position error statistics (TY only) 48
Fig. 22. Example PROLOG program 1. 58
Fig. 23. Example PROLOG program 2. 60

ACKNOWLEDGEMENTS

The support of several people was critical to the completion of this study. My thanks go to Prof. Russ Elsberry, who helped me define the problem and kept me properly focussed throughout my efforts. Paul Dobos provided hours of his time to help in the design and actual writing of the expert system, and was always available to discuss problems as they arose. He was also instrumental in providing the test set data used to validate the final algorithm. Jim Peak (Naval Environmental Prediction Research Facility) and Prof. Neil Rowe (Naval Postgraduate School, Computer Science Dept.) both provided invaluable assistance during my efforts to learn the PROLOG computer language.

I must also thank those people who were the sources of the data and other information that went into this project. CDR Herbert Colomb (Commanding Officer), LT Cecil Johnson (Executive Officer) and their staff at Naval Oceanography Command Facility, Cubi Point provided comprehensive storm observation data as well as information on the setting of Conditions of Readiness. CDR Colomb, LT Marsha Jones, LT Henry Jones and LT Mark Gunzelman provided valuable insight based on their experiences as Typhoon Duty Officers.

Finally, I must express my deepest gratitude to my wife, Ellen, and my daughter, Lauren. These two very special people endured the many hours I had to spend away from them and the moods I went through during those rare times when we could be together. Without their endless patience and support, this work would not have been accomplished.

I. INTRODUCTION

A. CURRENT NEEDS

"The tropical cyclone constitutes one of the most destructive natural disasters that affects many countries around the globe and exacts tremendous annual losses in lives and property. Their impact is greatest over the coastal areas...." (Elsberry et al., 1987) Due to their destructive potential, accurate forecasting of tropical cyclones is of prime importance to operational meteorologists. Efforts to improve forecasting capabilities have resulted in the introduction of many statistical (e.g., climatology, persistence, analog) and dynamical (e.g., National Meteorological Center's Moveable Fine Mesh Model, Fleet Numerical Oceanography Center's One-Way Tropical Cyclone Model) forecast aids for operational use. These are all aids that support, but cannot replace, the skills, reasoning and knowledge brought to bear on the typhoon forecast problem by the human forecaster.

As a forecaster observes tropical cyclones over a region for a period of time, he will develop a sense of how they behave. This intuition operates in a process similar to an internal analog model and can include ideas about likely development areas and typical precursors, predominant tracks, and the likely impact of a storm on local human activity. Additionally, he learns which forecast aid might be the most useful in a given situation for his geographic area, how specific local topography/geography might modify a storm and its effects, and what consequences will result from issuance or modification of a typhoon forecast. These factors are difficult, if not impossible, to include in the currently used statistical or numerical models.

The expert system is a newly developed tool that is specifically designed to deal with non-mathematical, conceptual information such as a forecaster's personal knowledge and intuition. Expert system languages such as LISP or PROLOG are designed to use a set of known facts and a set of rules to deduce new facts from which a conclusion ultimately can be reached. This declarative process allows for much more flexible computer programs than are possible using procedural algorithms as in standard PASCAL, FORTRAN or BASIC languages. Artificial intelligence (AI) techniques easily handle such practical problems as missing or contradictory inputs, and can be used to evaluate all possible conclusions that logically follow from the given inputs. The main reason to suspect the usefulness of AI in meteorology is this capacity to generate several possible

outcomes from a single set of data. However, this may simply reflect real situations in which there is more than one arguably logical forecast.

The ultimate, and thus truly expert, system would imitate the human forecaster's thought processes and thus provide a total tropical cyclone forecast. Such a system would have to consider all the objective guidance, as well as any additional dynamic information on the environment's effect on typhoons currently under study but not yet included in models. Examples might include typhoon interaction with terrain, other typhoons, mid-latitude or tropical upper tropospheric troughs, subtropical ridge or monsoon surge, or the extratropical transition process (Sandgathe, 1987). Such a system would also have to take into account the empirical rules of thumb based on the intuition and observations of the human forecasters at least to the extent that they can be defined.

B. THE NAVY TYPHOON PREDICTION PROBLEM

An expert system for typhoon prediction would be of particular use to the U. S. Navy, which faces some peculiar problems that are not necessarily encountered in the civilian sector. These problems make it critically important that Navy forecasters have available to them an accurate, well accepted aid to allow them to make confident, precise and timely estimates of a typhoon's effects. An expert system could help reduce forecast errors that could have strategic military consequences by providing the forecaster with a "second opinion" that is based on a consistent set of logic.

The first peculiarly Navy concern with typhoons is that nearly all Navy personnel, facilities and equipment operate in the coastal environment where typhoon damage is most severe. Civilian populations can be spread inland where typhoon effects are attenuated. In case of a typhoon strike, the entire Navy population of an area can be affected. It is vital to Navy, and thus national, interests to know in advance what the local effects of a typhoon will be since ships, docking facilities and naval stations will bear the brunt of the storm winds, waves and storm surge.

The Navy must specifically worry about storm effects at sea. Typhoons can have many serious effects on maritime operations, such as the damage or loss of ships and injury or death to crewmembers. Less devastating, but still significant, are the tactical effects of a typhoon. Ship and aircraft routing and task force integrity can be disrupted. The winds and rain can interfere directly with sensor operation, as well as human efficiency, which makes target detection and tracking difficult if not impossible. The typhoon can ensonify the water through the action of wind, waves and rain to such a degree that anti-submarine warfare efforts are severely disrupted, while the adversary

submarine can maneuver and transit relatively unaffected by even the strongest surface storm conditions.

Third, potential fiscal and political problems associated with Naval typhoon forecasts can cause great difficulty for the military forecaster. The meteorological forecast determines whether ships, planes and personnel must be evacuated, or if an entire base must be secured and closed down until the storm passes. Either course can result in lost or wasted man-hours of work, and the evacuation of a naval station or air base is very expensive in terms of fuel, administrative and other operating costs, which may deplete the budgets of the organizations involved.

Unit commanders are reluctant to order these expensive actions unless they are sure that they are prudent and necessary. This decision is the responsibility of the area commander, and is generally based upon the recommendation of his meteorologist. Even when a forecaster recommends a certain course, the area commander may be unwilling to fully act on that recommendation (LT Henry Jones, personal communication). That unwillingness may be based on military necessity or a lack of faith in the forecast. A proven expert system acting in tandem with the forecaster and confirming his recommendation might have the effect of strengthening that recommendation, and lead to a more prompt and appropriate response by the area commander. Time, effort and money could thus be saved through judicious use of a full-scale second opinion expert system. Ideally, the politics of the military weather warning system would be reduced, along with increasing the forecast accuracy.

The frequent turnover of personnel at the facilities is another unique problem in military weather forecasting. Standard tour lengths for Navy forecasters are on the order of two to three years. Unfortunately, it is not possible to keep someone experienced in tropical meteorology in the tropics on subsequent tours of duty. On the contrary, personnel are often sent to different environments to broaden their experience base. This means forecasters have to learn about the peculiar weather patterns in their new area of responsibility while on the job. Inexperience of the forecaster may contribute to bad forecasts. Furthermore, the frequent transfer of forecasters often represents a loss of expertise in the form of forecaster intuition and empirical rules that were developed during a tour, especially if these do not get passed along to the next forecaster. Such knowledge can at best be lost only temporarily while the new person rediscovers it, or at worst is lost permanently through this process.

C. THE EXPERT SYSTEM APPROACH

These military forecast problems might eventually be alleviated by advanced versions of the numerical prediction models. There are several reasons why an expert system would be a useful addition to the current system of numerical models. Advanced mesoscale numerical models are generally too complex and expensive to be applied at each remote forecasting site. Since versions of the AI language PROLOG are very inexpensive and can run complex algorithms on standard desktop computers, such a system could be made available at both shore and ship meteorological offices. For example, most Navy meteorological commands are already equipped with the Zenith Z-248 desktop computer.

PROLOG can be used to create an expert system that operates and interacts with the user in plain English. Corrections and additions to the program can be incorporated easily by local personnel with little special training. The structure of PROLOG allows for great program flexibility, and the forecast algorithm can be written to include an interactive mode.

Once standard knowledge and rule bases have been established, they remain in place for use in the forecasting process. The system can represent the available knowledge from many forecasters, which allows standardization of the forecast output regardless of who is generating the forecast. More importantly, as new knowledge is gained by the human forecasters at a location, it can be easily transferred to the expert system prior to the individual's departure, which will ensure that the accumulated expertise is not lost.

An additional benefit of an expert system is its usefulness as a training tool. To speed the learning process for a new forecaster, various input conditions can be entered into an expert system to illustrate the effects on the output forecast. The structure of the AI languages allows the user to interact with the system to trace the reasoning from the input information to the conclusion. This trace can include delineating what input and stored facts (e.g., synoptic conditions, storm location and strength) were used and what dynamical, statistical or empirical rules were considered. By seeing what data were used and how the decisions were made in the different trial scenarios, a novice forecaster can quickly get a feel for the local meteorological patterns as they relate to many tropical storms. All of these considerations lead to the conclusion that an expert system might be of great value to typhoon forecasting.

AI has already been applied to other meteorological forecasting problems with encouraging success. WILLARD (Zubrick, 1985) is an expert system to aid in severe thunderstorm forecasting, and AESOP (Peak, 1987) was designed to forecast visibility obscuration phenomena at sea. Peak (1987) points out that most expert system designers believe that constructing a limited prototype is the best way to begin development of an expert system, since it gives valuable information on the structure as well as the feasibility of the final system. For example, the first version of AESOP was only a prototype that illustrated a limited subset of its potential output.

The objective of this thesis is to produce a prototype of an expert system for the meteorological aspects of the setting of typhoon conditions at N.A.S. Cubi Point, Philippines. As indicated above, the setting of typhoon conditions requires consistent and accurate meteorological inputs. However, just the accumulation of the knowledge base in a convenient computer format for training purposes alone may justify the expert system.

D. PREDESIGN DECISIONS

The prototype system for typhoon forecasting designed in this study will operate as a stand-alone system, with the idea that it ultimately will become a small segment of a total typhoon forecast system. The approach is to start with currently available forecast guidance. After generating knowledge and rule bases and incorporating them in a PROLOG algorithm (this process will be discussed in detail in Chapter 2), AI techniques will be used to correct, or modify, the forecast for one or more of the seven types of environmental interactions (Sandgathe, 1987) that are not included in the model forecast guidance. These interactions generally have not been fully studied or analyzed, so the rule base will include empirical and statistical rules rather than dynamical facts.

Of the seven interactions that could be modelled using an expert system, terrain interaction was chosen for this feasibility study. This aspect of typhoon behavior has been examined in several studies, including statistically (Brand and Blelloch, 1973, 1974), numerically (Bender et al., 1987), and even in laboratory experiments (Wu and Wang, 1983). The emphasis here is on collecting and organizing the knowledge and rules, and writing the software rather than on generating the rules. This available knowledge on terrain interaction should result in more rapid prototyping and feasibility determination than from the other interactions listed by Sandgathe (1987) that have been studied less. In addition, Navy forecasters are specifically reminded in the Cubi Point Forecasters Handbook to "...be aware of the effects of topography in modifying tropical cyclones

when issuing forecasts and considering Conditions of Readiness", which emphasizes the continuing relevancy of terrain effects on typhoons.

Finally, Turbo PROLOG[1] was chosen to be the language of the system. This language is relatively simple to learn, can be commercially obtained for under $100, and will run on any of the Navy's IBM-compatible desktop computers. While the Turbo PROLOG version is missing some of the features of more complex versions, these omissions are inconsequential compared to the advantages gained, such as ease of operation and time saved in program development. Appendix A describes the Turbo PROLOG language and logic system in greater detail.

---

1 Turbo PROLOG software is a copyrighted product of Borland International, Inc..

II. DEFINING THE PROBLEM

A. PROTOTYPE SCOPE

In general, a limited prototype is the first step in creating any large model. In the case of a tropical storm prediction expert system, at least two fundamental factors make this step a necessity. First, the phenomenon to be modelled is complex. Second, the expert system concept needs to be validated for this application.

Due to the complexity of a tropical cyclone system, it would be impractical to attempt to create a comprehensive model right from the beginning. A vast amount of both empirical and dynamical information on typhoons is known, and this knowledge would have to be collected and translated into a form useable by a PROLOG programmer. In addition, the currently unknown factors summarized by Sandgathe (1987) need to be studied and characterized before they, too, can be included in a model.

The second problem in developing a total typhoon forecast expert system is that it has not been tried before. It is not yet clear if such an approach can be used, or if it can prove useful to forecasters operationally. The recommended technique to test the expert system concept is to get some limited version of the ultimate system programmed and running as early in the process as possible, then to test and expand on it as the appropriate form for a useable algorithm becomes clear.

Given these considerations, a prototype system is a reasonable goal for this study. However, the problem under study is complex enough that further restrictions are required to make it more manageable. Dealing with empirical and statistical rules for an expert system rule base creates considerations not encountered in constructing a dynamical model, namely locational dependence. Whereas a dynamical rule obeys the laws of physics and will apply at any location, empirical rules are different for the different geographical areas. To define the rule and knowledge base to be used in this study, a specific geographic site had to be chosen.

The initial candidate for the prototype expert system was Taiwan. It was considered on the basis of its distinctive, isolated terrain, and the frequency with which typhoons occurred there. Also, there is a large body of published literature that deals with the Taiwan-typhoon interaction problem numerically (Bender et al., 1987; Chang, 1982), statistically (Brand and Blelloch, 1974; Wang, 1980) and even experimentally (Wu and

Wang, 1983). It was assumed that a large body of rules could be generated to describe typhoon behavior around Taiwan based on the observations and results of these studies.

In addition to the rules available from published studies, direct input from the experts, the Western Pacific typhoon forecasters, were solicited in hopes of getting useful but unpublished rules of thumb. It was assumed that such empirical rules have evolved over time at each site, to be passed down from forecaster to forecaster. In a very early attempt to collect this input, an interview with former Typhoon Duty Officer LT Henry Jones indicated that Taiwan was not the best choice of a location for this prototype study. The main drawback is the lack of access to local forecasters at Taiwan. The main expert pool for this study consists of Navy forecasters, none of which, of course, have ever been to Taiwan to do forecasting. Instead, what little effort Navy forecasters placed on forecasting typhoon effects for Taiwan was done remotely from stations elsewhere in the Pacific. There is no first-hand experience available.

Researching the literature for another likely candidate revealed that the Philippines has been studied from a terrain-typhoon interaction perspective almost as extensively as Taiwan (Brand and Blelloch, 1973; Jarrell and Englebretson, 1982; Sikora, 1976). Due to the presence of a Naval Oceanography Command Facility (NOCF) at Cubi Point, there is a readily available source of forecaster expertise. Thus, there is a readily available source of the empirical rules necessary for a comprehensive expert system.

This prototype expert system hopefully will illustrate the usefulness of this concept as an aid in the Navy typhoon forecasting problem, both operationally and potentially for training. The U. S. Navy has considerable interest in typhoon forecasting for the Philippines. To demonstrate system applicability in a Navy-related location, the Subic Bay/Cubi Point complex was the choice for this study.

B. SELECTING THE GOAL

The next step in expert system generation is to define the goal of the system. As described in Appendix A, PROLOG programs find all available answers to a given question. This question can be included in the program if it is to be answered each time the program is to be run, in which case it is referred to as the goal. Alternately, the question can be input by the user at the start of each program run. Simply put, the answer(s) to this goal or question correspond to the output of the program.

Defining the desired goal may be the most difficult portion of the programming process. To come up with a concise, yet useful, goal requires a careful and complete definition of the problem before programming starts, since it is most efficient to start

with some definite goal in mind. With a limited and well-defined goal established, the rest of the program algorithm can be efficiently written.

Another consideration was that the expert system output should be in a form useful to the operational forecaster. There are any number of possible outputs that could be generated regarding typhoon-terrain interactions, let alone the alternate outputs from a comprehensive typhoon forecasting system. The aim of the project was to come up with a system that would mimic, albeit crudely, the forecaster's thought process. The system should accept as many sources of input as possible and incorporate them into one simple and concise forecast. The program should simplify the forecaster's task, rather than complicating it by adding more information that he must merge with the data already provided to him.

It was decided that a terrain-modified storm track and intensity were too complex to serve as the primary output. Based on forecaster interviews (personnel communications with LT Henry Jones, LT Marsha Jones, CDR Herbert Colomb), it was decided that the best way to characterize the typhoon effect on a station was by the locally observed winds (rainfall was not a variable under consideration for this study). This choice would reduce the parameters of the output to one. The problem of a large range of possible output values remained. Therefore, this goal was further reduced to include only a discrete set of wind speed ranges to be expected at Cubi Point, rather than the specific value of the wind speed.

This scheme of using discrete wind ranges was seen by forecasters in early interviews as having potential. They pointed out that a form of this structure was already in place. Whenever winds over 35 kt are expected within a certain period of time, a Tropical Storm Condition of Readiness (COR) is declared by the area commander (Commander, U. S. Navy Philippines, COMUSNAVPHIL). Precautions to minimize damage based on this condition are then taken. As the expected time until arrival of the maximum winds diminishes, higher COR's are set (Table 1). It was further pointed out by the forecasters that determining these COR's is the ultimate aim of local operational typhoon forecasters.

The goal for this prototype expert system is then to provide meteorological guidance for the setting of typhoon wind COR at Cubi Point This choice fulfills all the requirements for a good goal state. There are only four distinct possible responses to the query about an appropriate condition. This guidance for setting the COR value can be used, and its significance appreciated easily, by the operational forecaster without adding

complications to an already difficult forecast problem. The forecaster can then utilize this "second opinion" in formulating the meteorological inputs to the setting of the COR by the area commander, who must also consider strategic readiness, economic and human factors in his decision. Since it is this COR value alone that determines the precautions to be taken, it has tremendous practical and monetary significance to Naval operations.

Table 1. CUBI POINT CONDITION OF READINESS CRITERIA: Given the time until expected arrival of potentially destructive winds, this table shows the proper COR to set at Cubi Point based on local directives.

Time until arrival of winds over 35 knots	Tropical Storm Condition of Readiness
over 48 h	4*
24 h - 48 h	3
12 h - 24 h	2
less than 12 h	1
*Condition 4 is set continuously from 1 June to 31 December	

Using the expert system to generate COR forecasts has another benefit as well. This single parameter describes the ultimate and total effect of all dynamic environmental influences on the storm, at least with regard to how the result affects human activity. This prototype system will only attempt to account for terrain effects on storm motion and intensity, and on the resulting local winds. As more is learned about the other effects that determine how a typhoon behaves, this knowledge can be incorporated easily into this expert system, since the goal will not need to be changed.

III. GENERATING THE RULE BASE

A. INTRODUCTION

The general form of the program involves the data flow, rule base structure, and input and output requirements. Although general decisions on these aspects of the program are decided in conjunction with the goal selection phase, they are typically modified continuously thoughout the program writing phase.

The structure in Fig. 1 is the skeleton for the program. The finished program is much more complex than this figure indicates, and the terrain modifications of storm intensity and position indicated can be accessed though several different program options. However, this structure defines the rules needed in general terms, as well as some of the input and output requirements.

Three main functions are to be accomplished by the program: (1) Given a storm observation or forecast as input, generate a terrain-modified position; (2) Given a storm observation or forecast as input, generate a terrain-modified intensity; and (3) Given a terrain-modified or forecast storm position and intensity, determine the actual Cubi Point winds and associated tropical cyclone Condition of Readiness (COR). Each of these tasks requires a set of rules, either to prescribe how and when to apply terrain modifications or how to determine the Cubi Point winds given appropriate input. Sources (literature references are described below) for each of the three main rule groups are indicated in Fig. 1.

Many rules involved in the final algorithm are not part of one of these three main rule groups. These rules serve to manipulate the program data, do routine calculations and control program flow. The next three sections in this chapter will deal only with the translation of expert-derived rules into PROLOG. The other rules will be discussed in Chapter 4 where a detailed description of the overall program will be presented.

B. DETERMINING LOCAL CUBI POINT WINDS

The first rule group involves determining the expected winds at Cubi Point given the storm position and intensity. The sole source for the rules used in this study is Jarrell and Englebretson (1982), who calculated statistical relationships between storm parameters and the observed winds at Cubi Point.

Jarrell and Englebretson (1982) use storm data from 1955-1979, and select all best track data for storms within 360 n mi of Cubi Point. Wind observations at Cubi Point

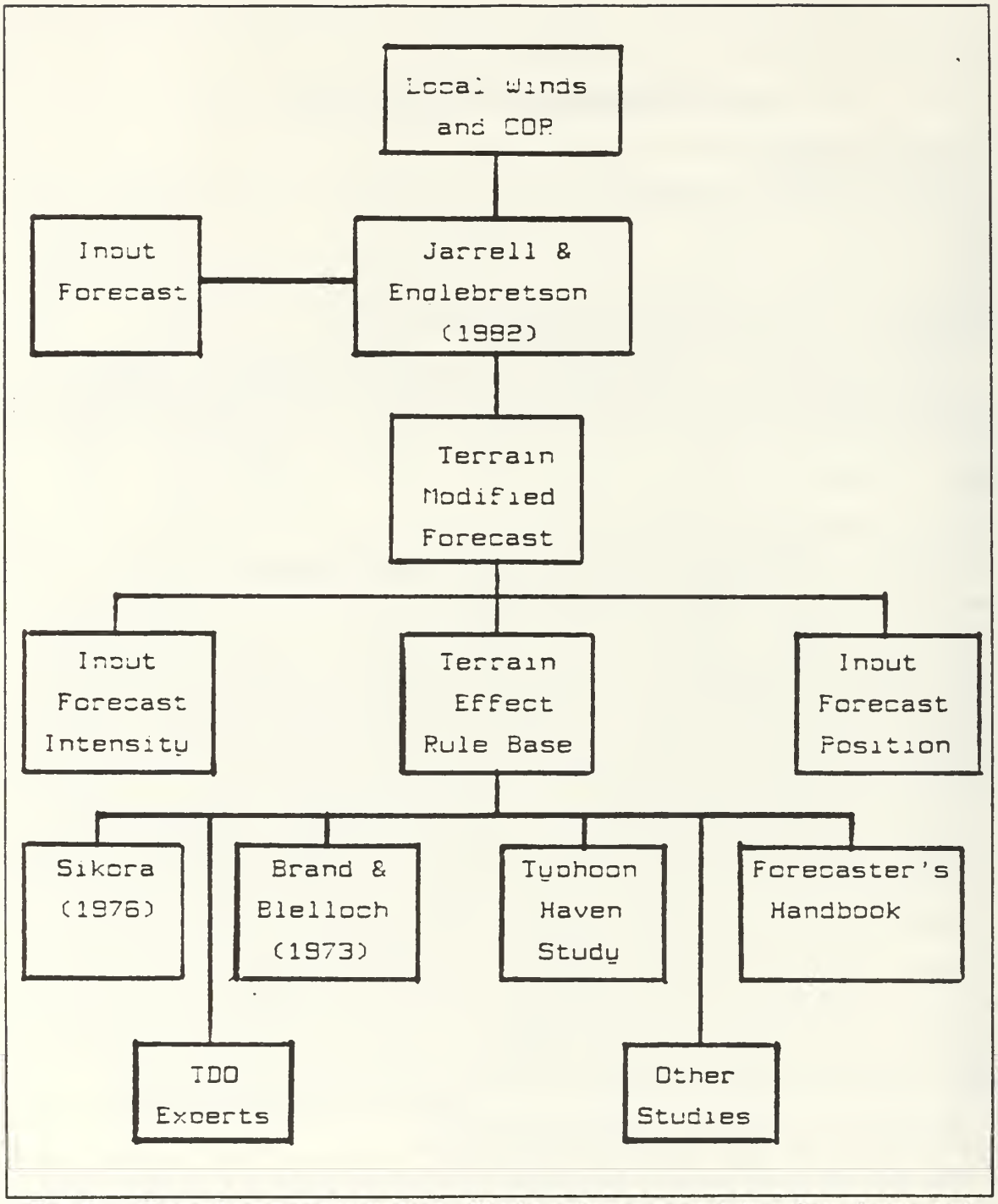

Fig. 1. General program structure: Blocks indicate general data flow. Input and sources for the rules are indicated. Output corresponds to the top block. Typhoon Haven Study refers to Douglass (1975).

were also collected corresponding to each of these best track data. These data were then divided into a set for storms having an intensity greater than or equal to 64 kt, and a set less than 64 kt.

Jarrell and Englebretson (1982) divide the 360 n mi domain into 71 equal areas as shown in Fig. 2. The maximum (average of the highest 10%) and mean (overall average) ratios between the storm maximum wind speed and the observed sustained wind observations was determined for each area segment for both strong and weak storms. These ratio values were provided in tabular form and also as contoured plots. A key assumption is that the maximum and mean ratios between storm intensity and observed gusts can be estimated by multiplying the maximum and mean sustained wind ratios, respectively, by 1.5.

The tabulated information from Jarrell and Englebretson (1982) used here consists of segment number (1-71), segment center latitude and longitude, and segment maximum and mean sustained wind ratios. A table is included for both strong and weak storms. All 142 of these table lines are included as facts in the program listing in Appendix B, for example *ts(44,[13.3,115.9] ,0.450,0.195)*. These facts then allow estimation of either maximum or mean gusts or maximum or mean sustained winds given the storm location and strength. The location of the storm relative to Cubi Point is used to determine the appropriate segment and associated ratio values. A more detailed explanation of how this process is accomplished in the final algorithm will be provided in Chapter 4.

C. INTENSITY MODIFICATION

The source for the rules determining storm strength modification is Sikora (1976), who studied all tropical storms and typhoons that crossed the Philippine islands during 1959-1975. The 6 h average intensity values from 48 h prior to landfall on the Philippines to 24 h after seafall into the South China Sea are plotted versus time. Different plots are included for storms north and south of 14.5°N and with strengths (averaged over 24 h prior to landfall) greater or less than 80 kt.

The resulting four curves (Fig. 3 and Fig. 4) represent the storm average intensities versus time to/since landfall. The intensity changes represented by these curves are derived from the straight line segments with endpoints as noted in Fig. 3 and Fig. 4. The slope, or fractional intensity change per hour, is calculated using the formula

$$\Delta I\,(h^{-1}) = \frac{I(t + \Delta t) - I(t)}{I(t) \times \Delta t} \qquad (3.1)$$

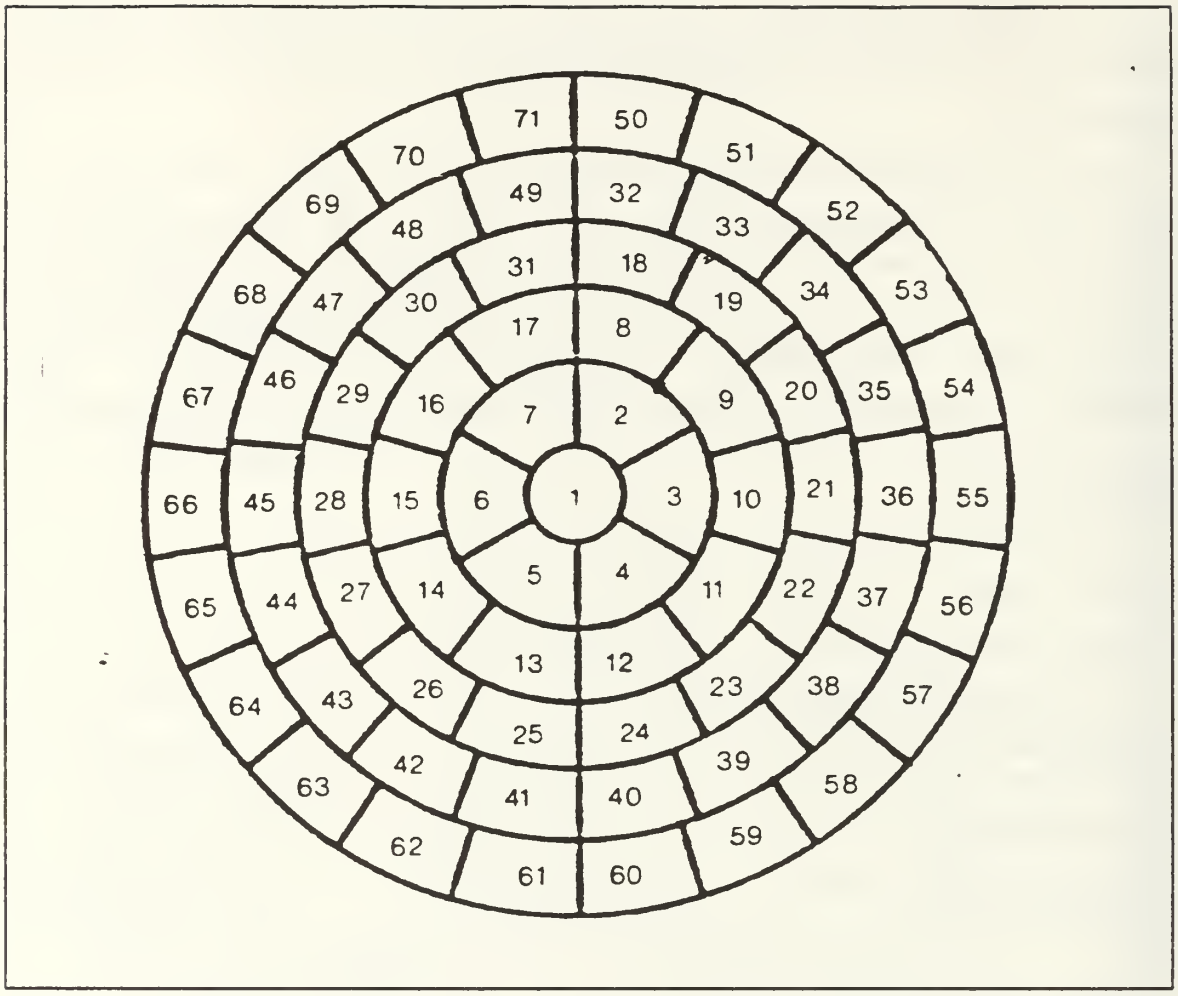

Fig. 2. Segments used in maximum and mean gust calculations: Outer radius is 360 n mi centered on Cubi Point. Segment areas are equal, dimensions are not constant. (From Jarrell and Englebretson, 1982)

for each of the four curves. The results comprise the *strength* rules in the program (see Appendix B).

These rules need input values of initial intensity, forecast interval being used, latitude and time to/since landfall. Initial intensity and storm latitude define which of the four curves is appropriate. Then, the time to/since landfall determines the line segment on the curve from which the slope is to be obtained. Finally, the modified intensity is calculated by multiplying the original intensity by a factor of (1 + slope × forecast interval).

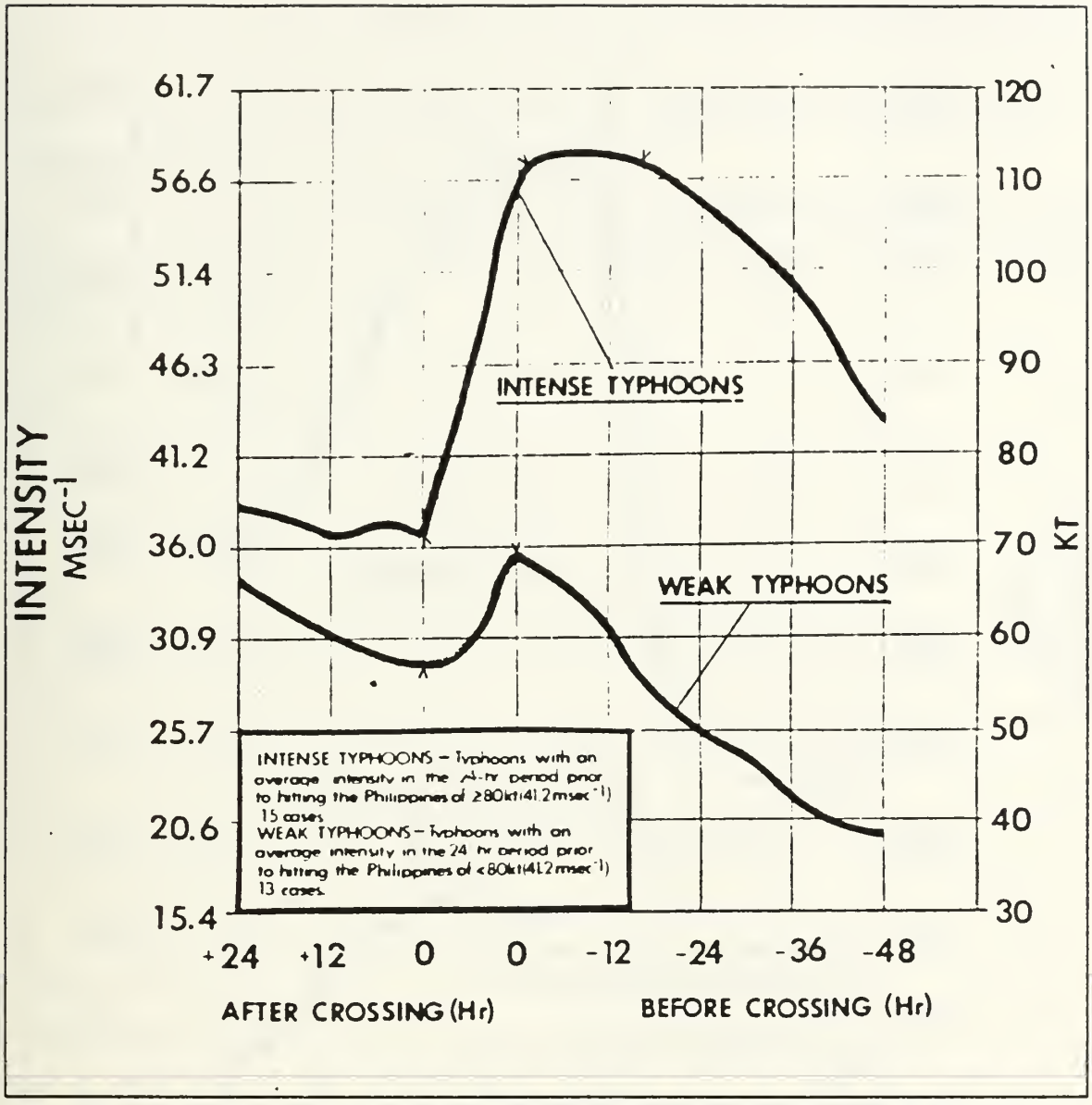

Fig. 3. Average intensity profiles: Average intensity versus time to landfall or since seafall for storms crossing the Philippines north of 14.5°N. Land is between the 0 times. Marked points indicate end points of straight line approximations used in the algorithm. (After Sikora, 1976)

D. POSITION MODIFICATION

The rules that govern position modification are generated from Sikora (1976) and Brand and Blelloch (1973). The modification process is divided into two steps. Along-track changes are calculated separately from the north-south modification.

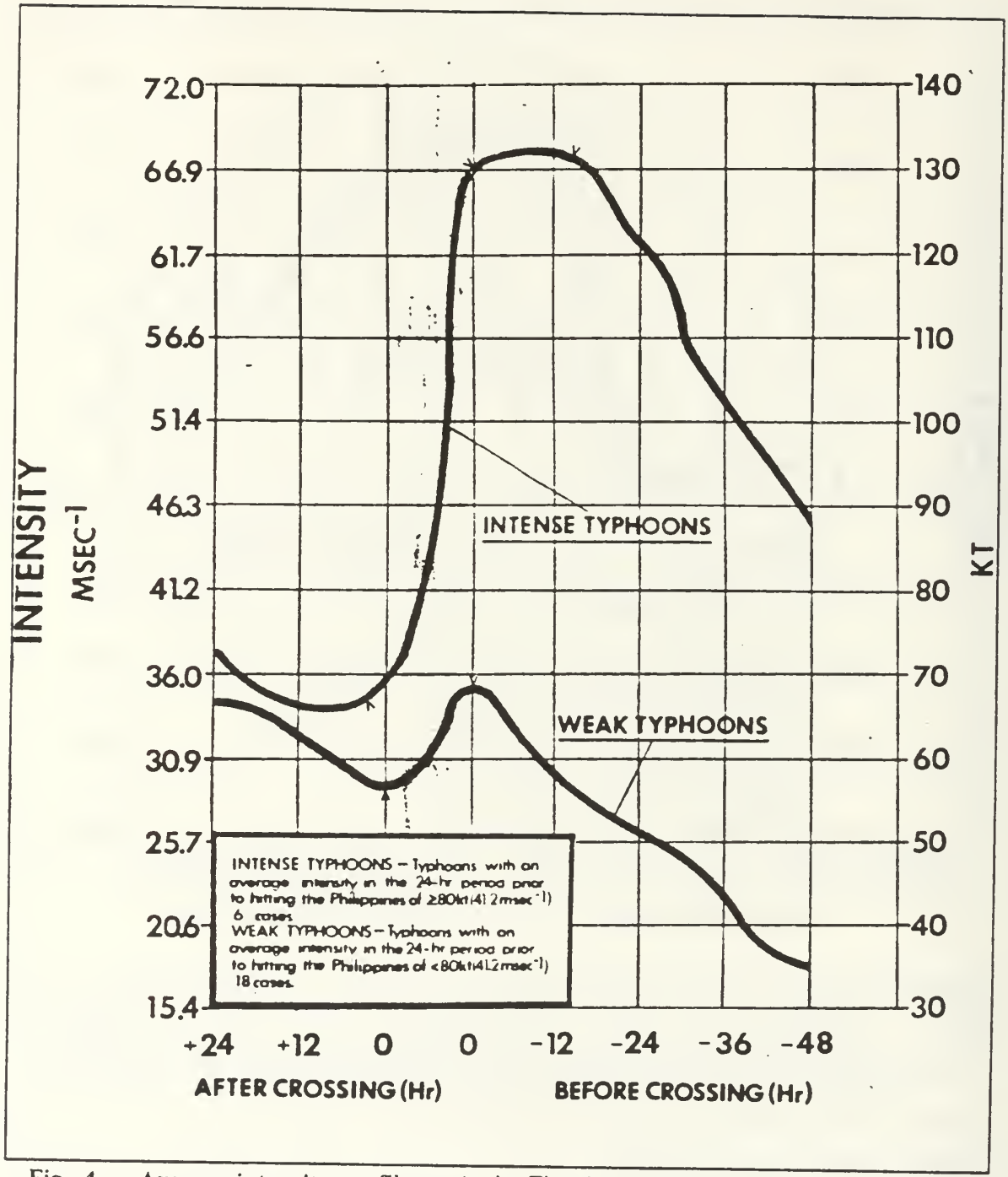

Fig. 4. Average intensity profiles: As in Fig. 3. except for storms crossing the Philippines south of 14.5°N. (After Sikora, 1976)

The along-track changes are a direct result of storm speed modifications. Translation speed modification rules were derived from Sikora (1976) in much the same way

as were the intensity rules. Sikora (1976) generated five curves representing average storm translation speed versus time to/since landfall for the cases of intense and weak storms, both north and south of 14.5°N. The extra speed curve is due to an observed bimodal distribution of weak southern storm speeds.

These five curves (Fig. 5 and Fig. 6) are divided into approximate straight line segments at the points noted. The slope is converted into fractional translation speed change per hour using the formula

$$\Delta S\,(h^{-1}) = \frac{S(t + \Delta t) - S(t)}{S(t) \times \Delta t} \tag{3.2}$$

This slope factor is calculated for each graph segment, and is then incorporated into the program in the *speed* rules (see Appendix B).

The inputs of unmodified intensity, forecast interval, time to/since landfall, latitude and unmodified speed are required for the *speed* rules. Latitude and intensity (and translation speed in the case of weak, southern storms) define the appropriate curve, and landfall time determines the segment on that curve. From this segment, a modification factor is calculated based on (1 + slope × forecast interval). This new factor represents the ratio between the modified translation speed and the observed speed. Unlike the *strength* rules, only this modification factor is returned, rather than the modified speed.

This translation speed modification factor is used to calculate a modified position and a modified speed in the *posit* rule. The new position is derived based on the assumption that any modification of the forecast storm translation speed must result in a corresponding change in the along-track travel of the storm. The modified travel distance is found without first calculating the modified speed by using the modification factor directly, such that

$$Mlat = Olat + Factor \times (Flat - Olat)$$

$$Mlong = Olong + Factor \times (Flong - Olong)$$

where: observed position = (Olat,Olong)

forecast position = (Flat,Flong)

modified position = (Mlat,Mlong).

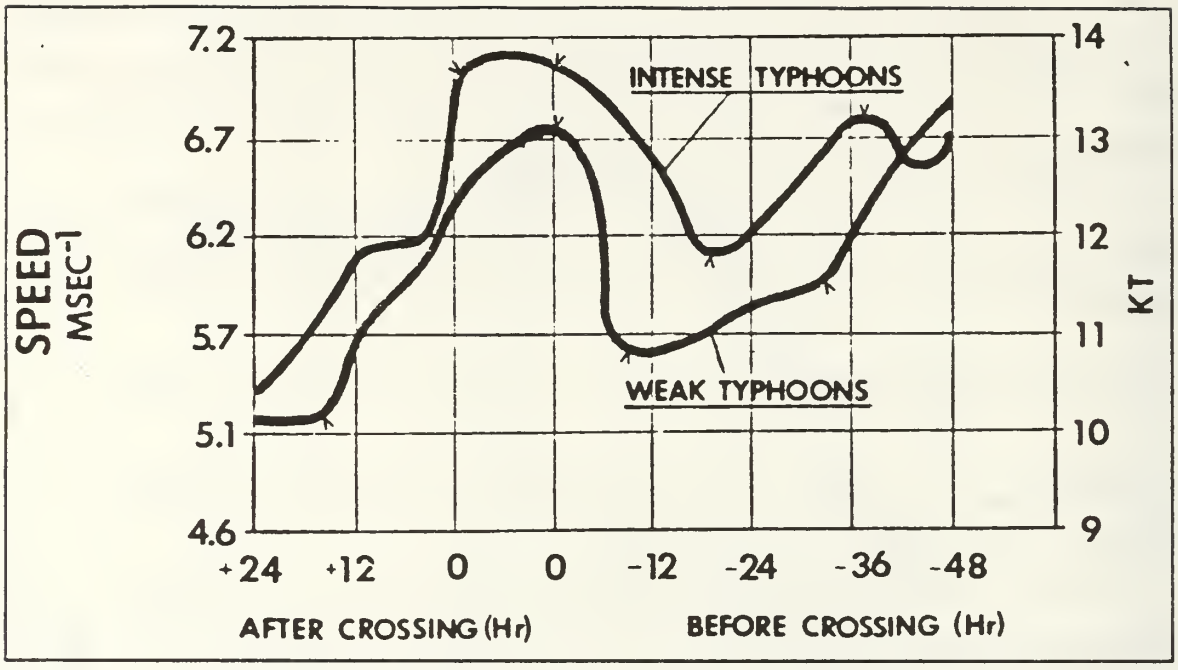

Fig. 5. **Average translation speed profiles:** Average translation speed profiles for storms crossings the Philippines north of 14.5°N. Land is between the 0 times. Marked points indicate the endpoints of straight line approximations used in the algorithm. (After Sikora, 1976)

That is, the ratio of modified to unmodified storm travel distance (radial, north/south, or east/west) equals the ratio of modified to unmodified speed, or the modification factor.

After the new position is calculated, a modified speed value is determined as the product of the observed speed and the modification factor. It is not used in the iteration in which it is developed, but may serve as data during subsequent iterations of the algorithm.

The position modifications are not complete at this point since the north/south component has yet to be included. The *xtrack* rules for this correction are derived from Brand and Blelloch (1973), which is the original study describing storm parameter changes during passage over the Philippines.

One aspect of Brand and Blelloch (1973) that has not been updated is their analysis of apparent cross-track (or north/south for these westward moving storms) forecasting errors (Fig. 7). The 18.3 n mi forecast error to the south for storm forecasts over land and the 14.1 n mi error to the north after land crossing are significant enough to be

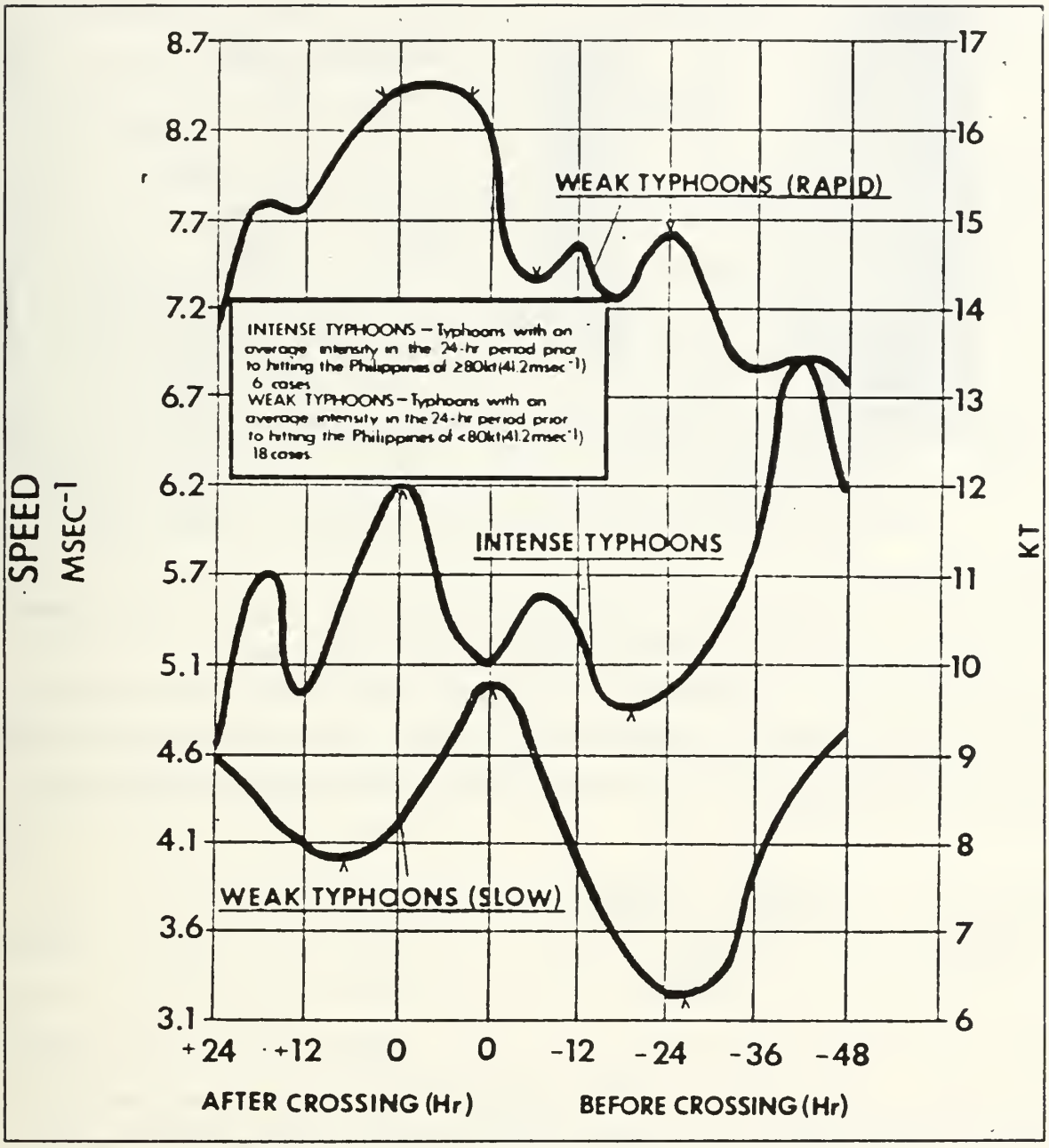

Fig. 6. **Average translation speed profiles:** As in Fig. 5, except storms crossing the Philippines south of 14.5°N. (After Sikora, 1976)

considered for this algorithm. Whatever the physical reasons may be for these errors, including them in the modification procedure is expected to enhance forecast accuracy.

Translating the values in Fig. 7 into PROLOG rules starts by assuming that prior to landfall, no correction is needed. Further, a total northward modification to the

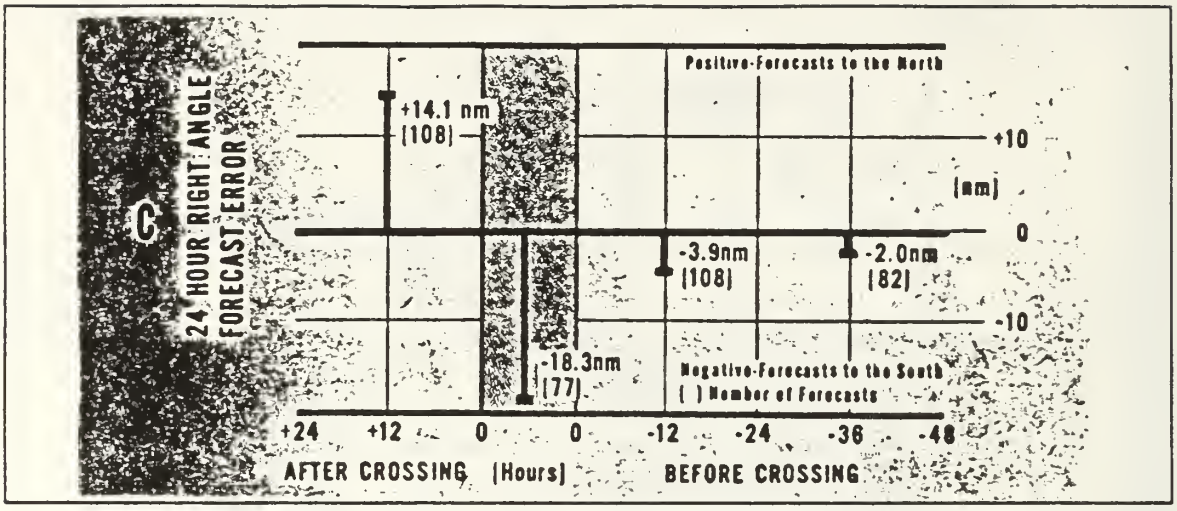

Fig. 7. Cross-track forecast errors: Average north/south forecast errors observed for storms crossing the Philippines. Land is between the 0 times. Negative error indicate forecast too far south. (After Brand and Blelloch, 1973)

forecasts of 18.3 n mi is assumed for transit across the Philippines, and a total southward correction of 14.1 n mi is assumed during the 24 h after crossing land.

It was necessary to ensure that the total correction is added to the forecasts no matter how many forecasts occurred in the over-land or 24-h post-land regimes. The solution is to allow the long-term average correction to equal the two appropriate values by using the relationship

$$\frac{\text{correction}}{\text{forecast}} = \left(\frac{\frac{\text{correction}}{\text{transit}} \times \frac{\text{hours}}{\text{forecast}}}{\frac{\text{hours}}{\text{transit}}} \right). \tag{3.3}$$

The values on the right hand side of (3.3) are either a fixed constant or are detemined by the input. Correction/transit is either 18.3 n mi or 14.1 n mi, and hours/transit was statistically calculated to be 13 h in Brand and Blelloch (1973). Although this figure was updated in Sikora (1976), the original figure is used in the algorithm to preserve consistency with the total correction figures, which were not updated.

The correction applied to a given forecast depends on the forecast interval and where the forecast falls in relation to land crossing. If the forecast falls over land and the forecast interval is such that the next forecast must occur over the South China Sea, the entire 18.3 n mi correction to the north is added. If the forecast interval is such that there is a chance that the subsequent forecast also will be over land, less than 18.3 n mi

is factored into this forecast modification. If the forecast interval is longer than 13 h, a check is made to see if the forecast was modified while over land. If not, 18.3 n mi is included prior to the appropriate post-land correction.

It is important to note that the result of this method is a long-term average correction equalling the values in Brand and Blelloch (1973). However, many cases will receive more or less correction over the time-iterated forecasts. In the second example above (forecast interval less than 13 h), there is no possibility of a storm receiving a correction of exactly 18.3 n mi to the north while over land. Consider a forecast interval of 10 h. If one forecast falls within three hours after storm landfall, the next must fall over land, sometime in the last three hours of transit, assuming a 13 h transit time as the algorithm always does. Each of these two forecasts would be modified to the north by an amount equal to 18.3 n mi × (10/13), as in (3.3), for a total correction of 28.2 n mi. If the initial forecast is not in the first three hours over land, but in the next seven hours, there would not be any possibility of another forecast over land. While the appropriate correction is 18.3 n mi, the correction here would be 18.3 n mi × (10/13), or 14.1 n mi.

The total correction applied averages out to exactly 18.3 n mi because the relative probabilities of the two events cancels out the over- and under-corrections. The chances of the over-correction occurring is 30% (the forecast must be in the first 3 h of a possible 10 h), while the under-correction occurs 70% of the time. Thus, the total average correction applied in the long run is 18.3 n mi × (0.7 × (10/13) + 0.3 × (20/13)) = 18.3 n mi.

This correction is applied after the along-track position correction discussed earlier. Once the magnitude of the north/south correction is calculated based on forecast interval and storm location with respect to seafall, it is added directly to the once-modified latitude as degrees north or south.

E. CLIMATOLOGICAL STORM TRACKS

Sikora (1976) presented five climatologically preferred storm tracks around the terrain of the Philippines (Fig. 8). To enable a user to determine the effects of a given storm following one of these paths, a set of rules defining the paths was written.

The first part of the definition process consists of generating the equations for the different tracks. The slope and intercepts are calculated using the path segment endpoints and assuming surface curvature effects over this small area are insignificant. The southern three paths each have a bend, so two sets of values are calculated. Given the longitude, the latitude of any point on the paths may be calculated.

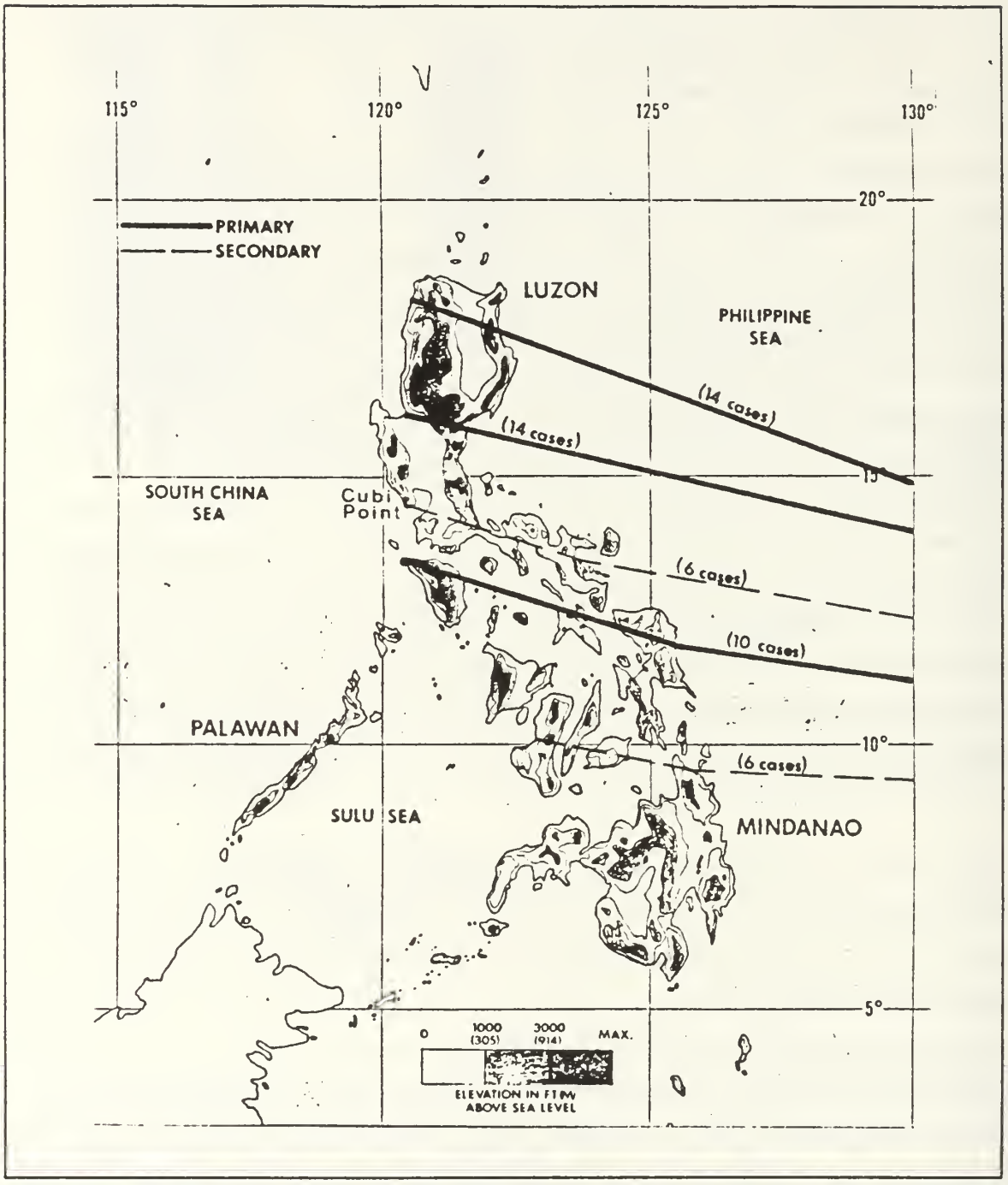

Fig. 8. Climatological storm tracks: Chart showing the climatologically preferred typhoon tracks across the Philippines. Solid lines indicate primary tracks, dashed indicate secondary tracks. Number of cases averaged to generate tracks is indicated beside each. (From Sikora, 1976)

Second, the true heading along any segment of the paths is calculated using the same assumptions. These values will be used to extrapolate a new path position given an old position and a storm travel distance. Along with the equations for the path points, these headings constitute a set of *path* rules in Appendix B.

IV. THE PROLOG ALGORITHM

The typhoon prediction algorithm in Appendix B initially asks the user to input a storm observation, including date and time, position, intensity, translation speed, and current Cubi Point winds and COR. Then, the main menu is displayed, with several options available (Fig. 9). The sequence of events initiated by choosing each option will be explained below. Although general terminology will be used to outline program operation, Appendix A should be read prior to this chapter to ensure familiarity with basic PROLOG concepts.

A. UNMODIFIED FORECASTS

Option 1 from the main menu initiates the sequence in which COR and local winds will be predicted without modification of the user input storm forecast position and intensity. This option is useful for comparing terrain-modified with unmodified predictions, or as a training aid to illustrate the effects of various storm locations and strengths. The user is asked to input these forecast data, which are redesignated as the modified forecast to generate data similar to that of the other options.

A call to the *condition* rule is then made. *Condition* first finds the maximum sustained winds expected at Cubi Point by accessing the *ts* or *tc* facts described in Chapter 3 for storms over or under 64 kt in intensity, respectively, as derived from Jarrell and Englebretson (1982). The program generates a list including all of the 71 segments whose centers are within 80 n mi of the storm position. Because no point in any segment is farther than 80 n mi from the segment center, this ensures that at least one segment will be in the list. The closest segment center to the storm position is then found from this list. The storm intensity is multiplied by the maximum gust ratio for this segment to get a worst-case estimate of the gusts at Cubi Point for the forecast time. The *condition* rule then calculates the time interval from the observation time to the forecast time. Based on the strength and time until arrival of the winds, the appropriate COR is determined from Table 1 and displayed. These data are stored for a later summary display of the entire run.

At this point, a second menu is displayed that lists the available options based on the original input data that are still in memory (Fig. 10). Option 2 will be explained in the next section. Option 1 can be repeated with a new forecast, or an iterative process can be initiated by selecting Option 3. This option saves the original observation time

Will you want to:

1) Determine COR based on your unmodified forecast,

2) Let the program modify your forecast and pick the condition,

6) Extrapolate observation along the closest statistically preferred path across the Philippines,

7) Extrapolate observation along a user chosen preferred path,

or 9) Exit the program?

Fig. 9. Program main option menu: This menu is displayed immediately after the initial storm observation data is input at the start of a program run.

input and bases COR forecasts on the interval since this time. The previous forecast is then redesignated as an observation and the program asks for a subsequent forecast input from which further predictions are generated as in Option 1.

Following the selection of Option 3, it is assumed that the unmodified forecast iteration process will be continued until the user is satisfied with the results. The only options available on further iterations are Option 3 and Option 9 to terminate the run. Appropriate ending conditions are a lack of further forecast input, forecasts that are outside the 360 n mi area around Cubi Point where the algorithm is not valid, or when the peak COR has been reached and only condition four is expected from subsequent forecasts due to weak winds or forecasts beyond 48 h.

B. MODIFYING FORECASTS

Selecting Option 2 from the main menu (Fig. 9) allows the user to apply all of the terrain modification rules to the input forecasts. After the original forecast is input, the *modify_forecast* rule is called to control the modification sequence.

The difference between the input observation time and forecast time is accessed. Using the input translation speed and assuming westward motion, the time to/since storm seafall is calculated, where seafall is defined as the time the storm crosses the 120.3°E longitude line if north of 14.5°N or the 122.0°E line if south of 14.5°N. If this time plus the average time for storm crossing based on latitude and storm strength (north: intense, 7.4 h; weak, 9.1 h; and south: intense, 26.4 h; weak, 18.4 h, from Sikora, 1976) indicates the storm has crossed onto land, the user is asked whether the average

Will you want to:

1) Determine COR based on your forecast,

2) Let the program modify the forecast and pick the COR,

3) Replace the observation data with your forecast and input a later forecast,

or 9) Exit the program?

Fig. 10. Secondary menu (after choosing Option 1): Example of the menu displayed showing the program options after Option 1 has been selected from the main or a secondary menu.

storm strength over the 24 h prior to landfall was greater than 80 kt. If the storm has not reached land, the average strength is left unknown.

Next, *modify_forecast* modifies the storm intensity using the rules described in Chapter 3. Forecast position north or south of 14.5°N and intensity and time to/since seafall are checked. If the strength is input as known, the appropriate curve for intensity modification is chosen. If the intensity is unknown, the weak storm curve is selected since it involves the smallest reduction in intensity and results in a worst case estimate of modified intensity. The modified intensity is then calculated as described in Chapter 3.

The *modify_forecast* rule then calls for the modification of the position and translation speed based on the observation intensity, translation speed and latitude using the rules described in Chapter 3. At this point, all modifications to the forecast fact have been accomplished to generate the modified forecast fact, and the position, intensity and translation speed of both are displayed for comparison. The COR appropriate for the modified forecast is also calculated and displayed.

A new menu then appears (Fig. 11). Option 2 can be repeated using a new forecast, or an analog to the iteration process of Option 3 that now includes terrain modification can be selected by choosing Option 4. In this case, the modified forecast is redesignated as the new observation, and the subsequent user input forecast is modified as appropriate using the methods described above (see Fig. 12). Option 4 can be repeatedly selected, with the same procedure being followed each time. The COR's are based on the

interval from the initial observation time until the most recently input forecast. Option 4 does not allow the position modifications to be accumulated, so the modified forecasts will always be very close to the forecasts. This turns out to be ideal for testing the algorithm (see Chapter 5), but is not practical for forecasting since the storm position forecasts are never modified significantly for terrain effects.

The third option available after initially selecting Option 2 is the modified-persistence option (Option 5), which accumulates the full effect of the terrain modifications. Since Option 2 must precede Option 5, the program already has in memory the original observation, forecast and modified forecast. In Option 5 (Fig. 13), the heading and distance between the observation and the modified forecast is calculated and extrapolated to a new persistence-based forecast position. The previously modified forecast is redesignated as the observation, the new persistence forecast is modified for terrain effects and the resulting COR is displayed. Option 5 can be repeated until ending criteria are achieved. The only inputs that are needed are the initial observation and the first forecast required in Option 2. The remainder of the resulting storm track and the COR's predicted from it are solely the result of the cumulative terrain-induced modifications to a persistence track (see Fig. 13). The Option 2/Option 5 combination is designed as the primary forecast tool provided in the algorithm.

C. CLIMATOLOGICAL TRACKS

Options 6 and 7 in the main menu (Fig. 9) are very similar in that they both force the storm forecasts to lie along one of the five climatologically preferred storm tracks shown in Fig. 8. The storm intensity and translation speed are modified for terrain effects as above, but the positions must lie on one of the preferred paths. The difference between the two options lies in how the path is chosen. Option 6 selects the closest of the five paths to the input storm position. The closest path is based upon the north/south (latitudinal) distance from the observation to each path. Alternately, Option 7 allows the user to select any one of the paths for the subsequent calculations.

Once the path is selected by one of these methods, the input latitude is shifted directly north or south to the chosen path. The forecast process is then identical to that described previously. A forecast position along the chosen path is generated at a distance equal to the input translation speed times the forecast interval. All parameters of this forecast are modified for terrain effects as appropriate, but the modified position is again moved directly north or south to lie on the path. Then the position, intensity,

Will you want to:

2) Let the program modify the forecast and pick a COR,

4) Replace the observation with the modified forecast and input a later forecast to be modified,

5) Generate a persistence plus modification track with no forecast input,

or 9) Exit the program?

Fig. 11. Secondary menu (after choosing Option 2): Menu that is displayed showing available program options after Option 2 has been selected from the main or a secondary menu.

translation speed and COR are displayed. The process is repeated using Option 8, which bypasses the path selection process.

These options are particularly useful if there is any reason to suspect a storm will follow a particular path around the Philippines terrain. In the future, rules may be generated that will allow the program to make this decision internally. Also, different paths can be examined for the same storm observation to determine the different effects on Cubi Point.

D. EXITING THE PROGRAM

Choosing Option 9 at any point terminates the program run. A display recaps the results of each step accomplished, including the time, position, intensity, Cubi Point winds and resulting COR based on the forecast positions (Fig. 14). From this display, trends in storm parameters can be analyzed. The recommended COR to be set at the time of the initial observation will be the highest one achieved during the run.

At this point, the user has the option to save the summary display on floppy disk. This feature can be useful when comparisons are desired between runs based on successive operational inputs that are separated by 6 h, or between different options for the same storm. The user is prompted to input storm and run type identification, and the results are stored in a PROLOG file for later retrieval.

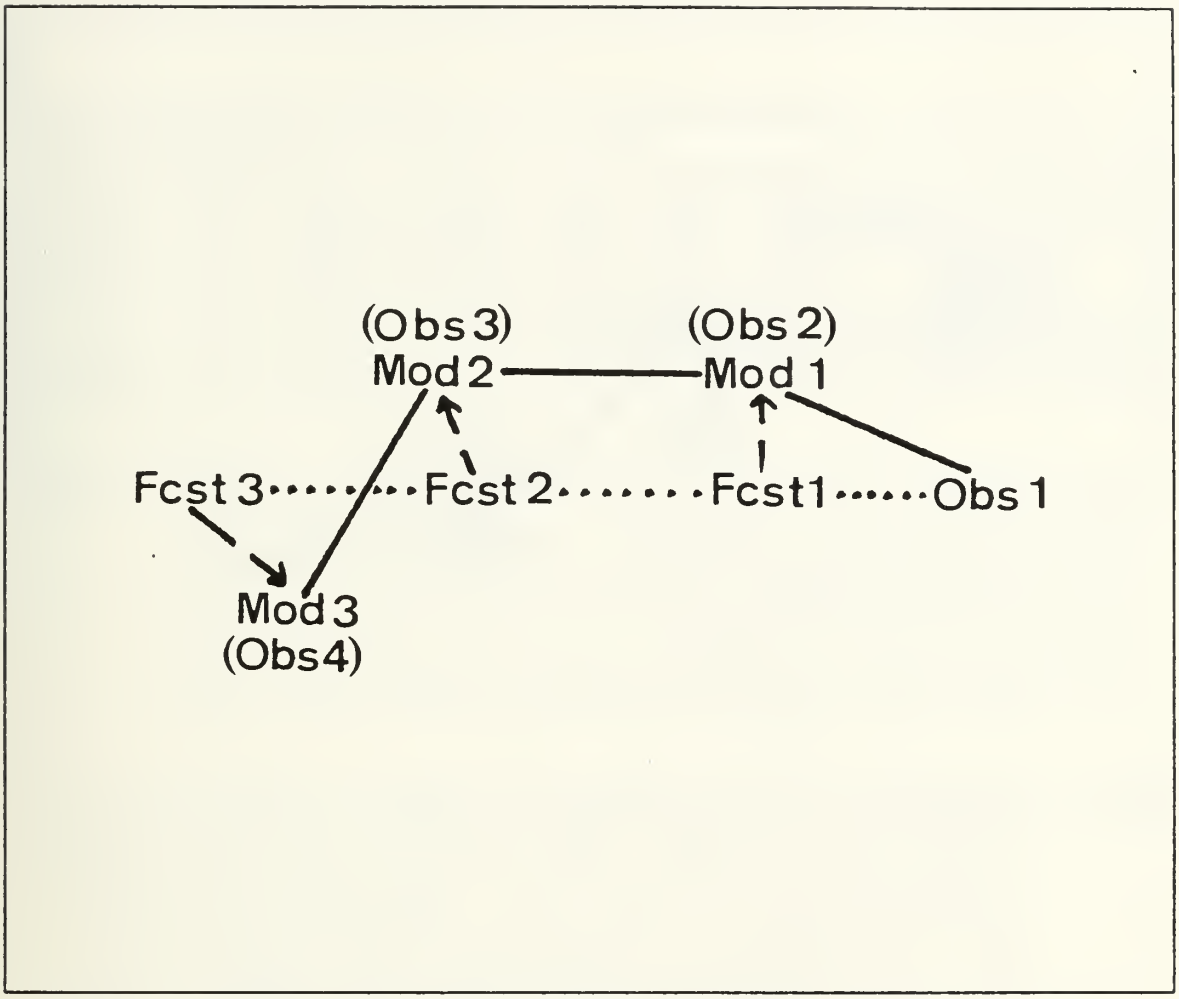

Fig. 12. Illustration of the Option 2/Option 4 procedure: Schematic showing the relationship of the original forecasts and terrain-modified forecasts in Options 2 and 4. Storm track before terrain influences are introduced is from Obs "N" to Fcst "N". Forecast track is along dotted line. Modifications are indicated by dashed arrows. Track including modifications is along the solid line. Terrain modifications do not accumulate. This is the procedure used to test intensity and speed modifications (see Chapter 5).

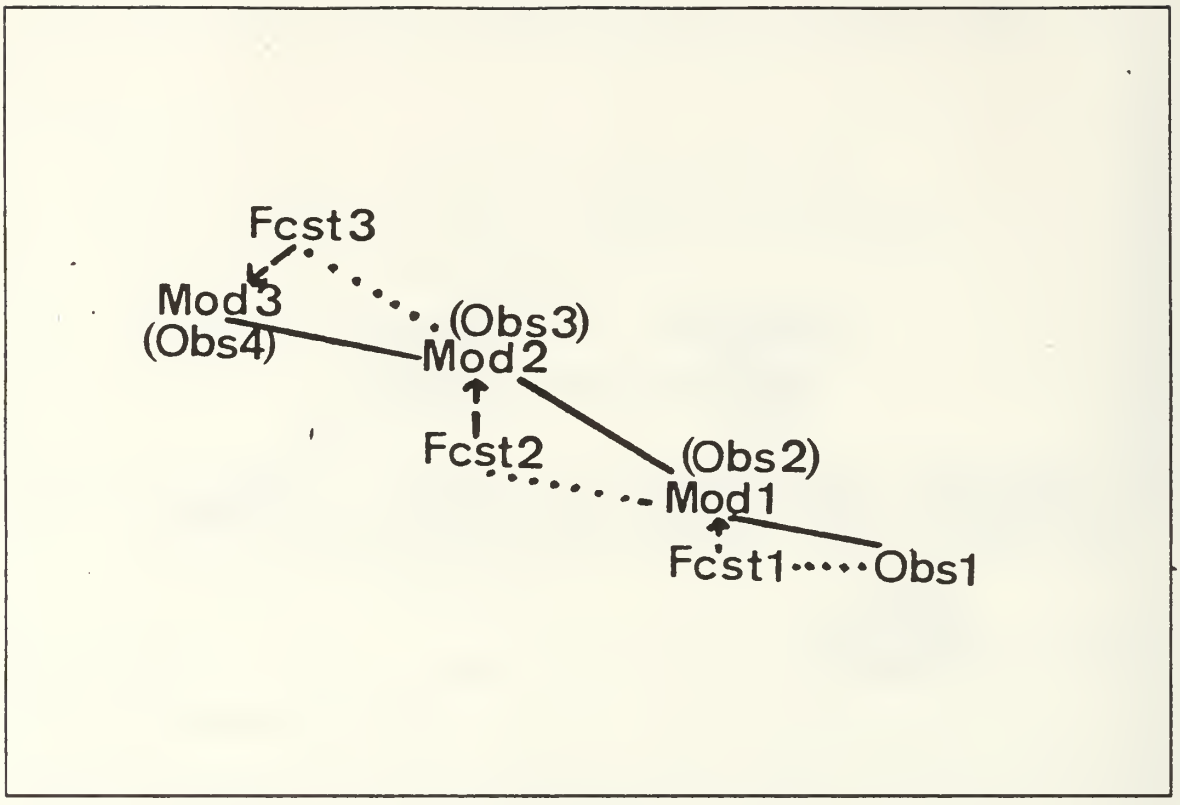

Fig. 13. Illustration of the Option 2/Option 5 procedure: Schematic of the relationship between the input forecasts and the terrain-modified forecasts in program Options 2 and 5. Segments between Obs "N" and Mod "N" is copied from Mod "N" to Fcst "N+1". Terrain modifications are indicated by dashed arrows. Forecast track segments are along dotted lines. Final track including terrain modifications is along the solid lines. Intensity, translation speed and position modifications are allowed to accumulate. This is the procedure used for testing position modifications (see Chapter 5).

ELLEN Option 2 5

Time	Day	Lat	Long	Int	Speed	Gust	Resulting Condition
600	12	13.20	120.00	40.00	12.30	99.00	Current TC cond.
1200	12	13.81	119.33	41.96	11.76	49.90	condition 1
1800	12	14.34	118.73	44.02	11.25	53.10	condition 1
0	13	14.78	118.18	46.18	10.75	23.18	condition 4 (winds)
600	13	15.15	117.70	48.21	10.28	24.21	condition 4 (winds)
1200	13	15.44	117.27	50.32	9.82	25.26	condition 4 (winds)

Fig. 14. Post-run summary display screen: A typical example of the screen at the termination of a program run via Option 9. Each line represents the forecast generated during one program iteration. Initial line is observation data, 99.00 kt is a place-holder only. Program recommended COR is the highest achieved, in this case condition 1. If the interval from the initial observation to a forecast is greater than 48 h, the Resulting Condition column indicates "condition 4 (time)". Gust values are the maximum wind gusts expected given the storm position and intensity.

V. TESTING THE EXPERT SYSTEM

A. DATA SET

To test the prototype expert system algorithm, a subset of the storms that occurred in the western North Pacific is selected. The first criterion is that each storm must have come within 360 n mi of Cubi Point so that the forecast aid for wind estimation of Jarrell and Englebretson (1982) will apply.

To test the various functions of the algorithm separately, a set independent of the data used to derive the relationships in this technique is needed. For example, Jarrell and Englebretson (1982) used the most extensive data set of the three papers (all storms from 1955-1979). Therefore, the second criterion is that only storms in 1980-1987 are considered for the test data set.

The personnel at the Naval Oceanography Command Facility (NOCF), Cubi Point, provided a comprehensive summary of storms affecting Cubi Point. The summary consisted of working positions for all storms from 1970-1987, as well as hourly wind and gust observations for the 49 h centered on the time of each storm's closest point of approach (CPA) to Cubi Point. The maximum pre- and post-CPA winds were included as well. See Fig. 15 for an example of the relationship between each of these winds. Unfortunately, local winds were not included for TY Ike and TS June in 1984 and TY Andy in 1985, which were therefore omitted from the sample.

The independent sample storms need to conform to the same requirements included in the source papers for the tests to be fair. The most restrictive of these is in Brand and Blelloch (1973), who analyzed the effect of terrain on non-recurving storms only. They only included storms that did not dissipate over the Philippine Islands, and that at some point reached typhoon strength. Further, Sikora (1976) dealt only with storms that travelled east to west over the Philippine Islands.

In choosing storms that meet all of the above criteria to generate a valid independent test sample, consideration has arbitrarily been given to the most recent storms first. In addition to typhoons, tropical storms that meet all criteria (except for reaching typhoon strength) are included in the test set. Results are analyzed with and without these weaker storms to determine the sensitivity of the rule base to storm strength. A final sample of 18 storms is selected from the 1984-1987 storms (Table 2).

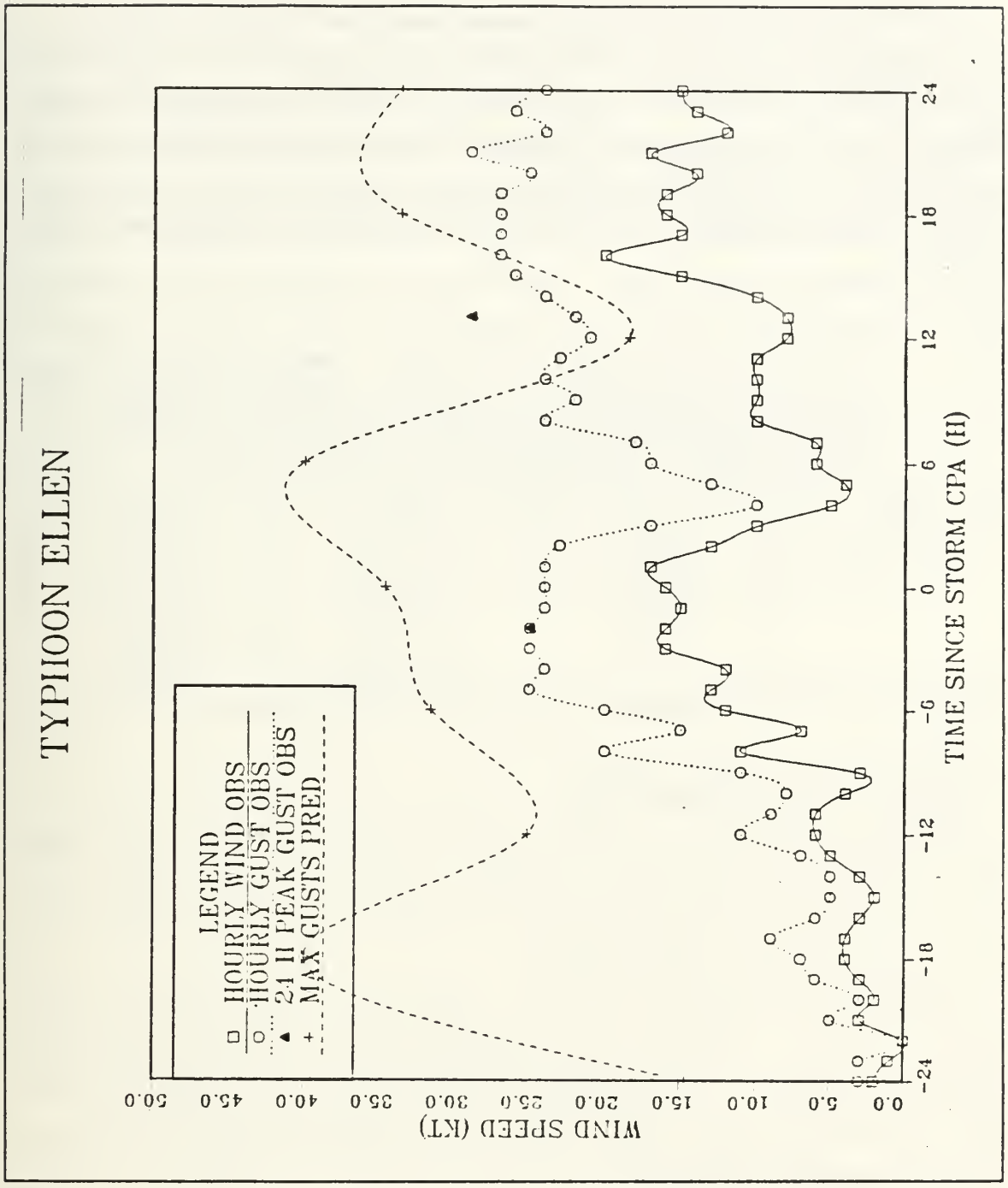

Fig. 15. **Relationship between wind types:** Hourly winds and gusts are from logged observations, and correspond to mean sustained winds and mean gusts. Peak winds are also observations taken in the 24 h prior to and after storm closest approach to Cubi Point, and equate to maximum gusts. Predicted winds and gusts are calculated each 6 h and represent maximum sustained winds and maximum gusts.

Verification data included *in situ* wind measurements for Cubi Point during storm passage obtained from NOCF Cubi Point weather logs. Best track and best intensity storm data were taken from the 1984-1987 Annual Tropical Cyclone Reports (ATCR) published by the Naval Oceanography Command Center/Joint Typhoon Warning Center (JTWC). Post-storm analysis positions, intensities and speed errors for the 24, 48 and 72 h forecasts issued each 6 h by JTWC are available from computerized JTWC error files.

Table 2. STORMS USED TO TEST ALGORITHM: Table listing storms chosen as test sample. Criteria include: east-to-west Philippine crossing, within 360 n mi of Cubi Point, not recurving or dissipating over the islands, and classified as typhoons. Tropical storms are included in some tests. TY = Typhoon, STY = Super Typhoon, TS = Tropical Storm.

1984	1985	1986	1987
TY Ike Ty Agnes	TY Hal TY Tess STY Dot TY Faye	TY Nancy TY Peggy TY Ellen TS Georgia TS Ida TY Marge	TY Betty TY Cary TY Lynn TS Maury STY Nina TY Phyllis

B. GENERAL TEST PROCEDURES

Since this study is the first to combine tropical cyclone information using an expert system framework, each aspect of the algorithm is tested separately. The objective is to determine any suspect areas in the rule base so that better rules can be generated to increase the overall accuracy of the algorithm.

The system accomplishes four main tasks. The first three tasks involve the modification of a forecast to generate a terrain-modified position, maximum intensity and translation speed forecast. The fourth task generates the expected local Cubi Point winds based on the distance to the (terrain-modified) storm location and maximum intensity.

C. WIND SPEED ESTIMATES

1. Test Procedure

The first and simplest test is to examine the algorithm derived from Jarrell and Englebretson (1982) that estimates actual Cubi Point wind speeds given the storm location and intensity. This test is done with the actual location and intensity rather than the terrain-modified values, which are evaluated separately below.

Program options 1 and 3 are utilized for this test. As these options were originally designed to provide maximum expected gust speeds (maximum gusts) from unmodified forecast data, no changes to the operational version of the program were necessary. Instead of inputting storm forecast data, 6 h best track and best intensity values are used. Given this "perfect" input, an assessment of the validity of applying the Jarrell and Englebretson (1982) rules to generate wind and tropical cyclone Condition Of Readiness (COR) estimates can be determined. Since Jarrell and Englebretson did not eliminate tropical storms from their data base, the three tropical storms in this sample (Table 2) are included.

From the Cubi Point weather log data, wind gust measurements taken during the hourly observations are available, if observed, for the time period from 24 h before to 24 h after storm CPA. Extracting the winds at 00, 06, 12 and 18 UTC (when storm data are available) gives at least eight winds (nine in the cases where CPA was noted at one of these times) per storm passage for comparison to program forecast estimates.

2. Wind Speed Results

Four types of winds may be predicted by the forecast aid, namely maximum and mean gusts, and maximum and mean one-minute sustained winds. To generate a worst-case scenario, the algorithm was designed to return values of maximum expected gusts, although a simple conversion using a 0.67 factor can be applied to get maximum expected sustained winds. However, output that most closely matches that which is logged by the observers in Cubi Point is the mean one-minute sustained winds and mean gusts (personal communication with LT Cecil Johnson, Executive Officer, NOCF Cubi Point). This difference will naturally make comparison difficult. Several types of comparisons will be conducted to determine the effectiveness of maximum gust or maximum sustained wind predictions for estimating actual wind values.

A summary of the errors incurred in predicting the wind speeds is shown in Table 3. Notice that forecast times and intervals are not involved since the wind calculations are on based on a series of nowcasts using best track storm center and best intensity data for the estimates. The values in Table 3 were calculated using the following formulae:

$$\text{Ave. Error Magnitude} = \frac{1}{N}\sum^{N} |X| \qquad (6.1)$$

$$\text{Ave. Error Bias} = \frac{1}{N}\sum^{N} X \tag{6.2}$$

$$\text{Error Std. Dev.} = \left(\frac{N \times \sum^{N} X^2 - \left(\sum^{N} X\right)^2}{N \times (N-1)} \right)^{1/2} \tag{6.3}$$

where X = Predicted - Actual wind.

Table 3. WIND PREDICTION ERROR STATISTICS: Magnitude, bias and standard deviation calculated for differences between predicted wind type values and observation values. Several prediction options are compared to observed gusts and sustained ("Sust") winds. Bottom row indicates optimum results achieved when a statistical tuning factor is included in the predictions.

Comparison between Predicted	vs Actual	Average Error (kt)	Average Error Bias (kt)	Average Error Std. Dev. (kt)	Sample Size
Max Sust (all)	Hourly Sust (all)	16.3	16.1	16.5	127
Max Sust (day)	Hourly Sust (day)	15.7	15.4	10.5	60
Max Sust (night)	Hourly Sust (night)	16.8	16.8	10.6	67
Max Sust	Hourly Gusts	10.3	7.4	8.7	73
Max Gusts	Hourly Gusts	20.6	20.1	14.5	73
0.28*Max Sust (all)	Hourly Sust (all)	4.4	-1.3	3.8	127

The average error of 16.3 kt when comparing predicted maximum sustained winds to the hourly sustained wind observations (Table 3) indicates that the worst-case estimates from the Jarrell and Englebretson algorithm seriously overpredict observed gusts. The fairly large positive bias indicates that most of the average error occurs from an overestimation of the actual winds. The high standard deviation shows that there is a large randomness factor in the estimates as well. In summary, most of these differences can be attributed to the use of a worst-case estimate of maximum sustained winds from the Jarrell and Englebretson (1982) statistics versus mean winds observed on the hourly intervals. The maximum ratios only apply in the worst 10% of all storm cases, and even in those cases they would not necessarily predict the maximum winds occurring during the hourly observations.

Since the actual local winds are measured in the boundary layer and might be subject to diurnal effects, the estimates and observed winds are divided into day and night groups. Observed winds at 00 and 06 UTC (08 and 14 Local) were designated daytime, and 12 and 18 UTC (20 and 02 Local) were designated nighttime. The results in Table 3 indicate only very slightly better results for the well-mixed daytime and only slightly worse results in the nighttime compared to the overall figures. However, these differences are not significant and do not explain the bias. This is not too surprising, since Jarrell and Englebretson (1982) used data from all times in deriving the figures used in generating the rules.

Comparison between the predicted maximum values of sustained winds and observed values of hourly gusts (when available) were conducted (row 4 of Table 3). The predicted maximum sustained winds show a much better correlation with these hourly gusts than with the hourly sustained winds. This improvement is attributable to a reduction in the overprediction tendency simply by comparing the predicted values to the hourly gust values, which by definition are larger than the hourly sustained winds. As with the previous comparisons there is a high standard deviation in these results.

The most critical test of the algorithm rules involves the comparison between the program predicted maximum gust speed values and the observed hourly gusts when noted. The results of this comparison (row 5 of Table 3) indicate a serious overprediction bias and a significant error magnitude of 20.6 kt. The bias is also evident in Fig. 16.

Recall that Jarrell and Englebretson (1982) derived ratios between the observed sustained winds at Cubi Point and storm maximum winds (see Chapter 3). Due to the maximum sustained wind values being serious overestimates as noted, a tuning factor that minimizes the error magnitude in this sample by converting maximum sustained wind estimates to mean sustained wind estimates was calculated independent of the mean ratios in Jarrell and Englebretson (1982). The statistics associated with this optimum factor of 0.28 are given in row 6 of Table 3. Notice that the optimum factor reduces the average error magnitude by half, while also reducing the bias magnitude and the error standard deviation. The optimum factor will not be incorporated at this time since the data set includes no storms that produced winds in excess of 35 kt at forecast times. Accuracy above the 35 kt value is more important than below it, so a sample including storms of this type must be studied before any changes are made.

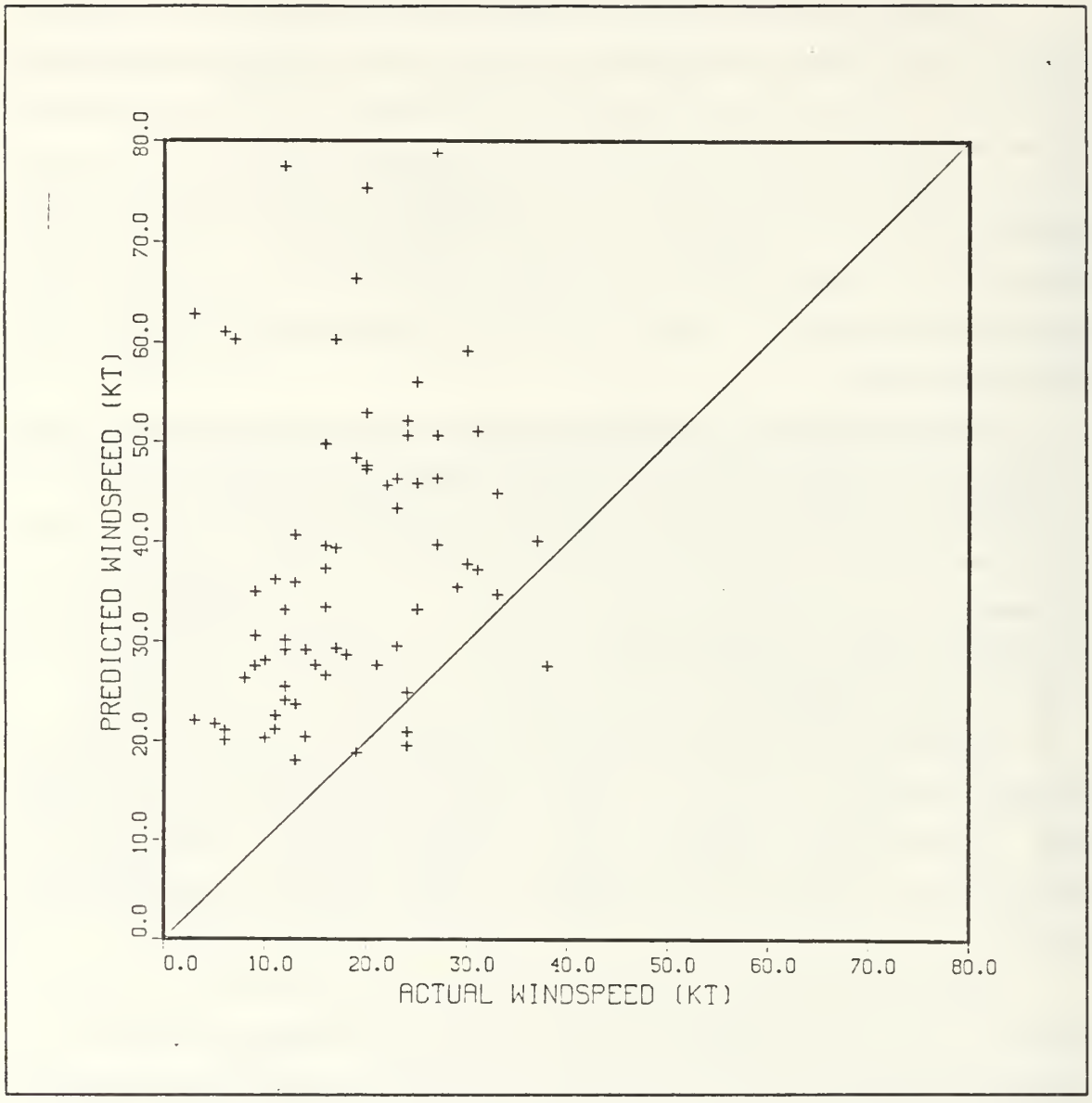

Fig. 16. Scatterplot indicating wind forecast errors: Predicted wind is maximum gusts expected at Cubi Point generated by the algorithm using best track position and intensities. Actual winds are hourly gusts at Cubi Point taken from wind observation logs.

The results in Table 3 are expected due to the selection of the worst-case maximum gust ratios from Jarrell and Englebretson (1982), and they do not indicate a fundamental weakness in the program. While there is only a poor correspondence between the program predicted wind values and the hourly observations, largely due to the use of the maximum wind ratios and the variability of winds at Cubi Point given similar

large-scale storm conditions (personnel communication with J. Jarrell), the program is not necessarily inaccurate overall. Certainly the results indicate more study into the local winds is needed for this phase of the program to be more accurate, but the main function of the program is to estimate the appropriate COR for Cubi Point. COR is based on the maximum expected winds, which correspond to the peak observations (see Fig. 15), but rarely coincide with the hourly observations. However, direct comparison between program maximum gusts and observed peak winds is only possible when the peak wind occurs within one hour prior to a 6 h best track data time. For the test storms used, this only happened one time (TY Phyllis post-CPA peak of 14 kt coincided with a predicted value of 9.1 kt). Therefore, comparison with the peak winds recorded will be done in the position modification section, when overall COR predictions will be generated instead of wind values for specific times.

D. INTENSITY AND SPEED MODIFICATIONS

1. Test Procedure

To test the validity of the terrain-induced intensity and storm speed modifications, program options 2 and 4 are used (Fig. 12). These options were originally designed for operational use and were only slightly modified for testing. These options require an observation and a forecast of storm location, intensity and speed. After modifying the forecast for terrain influences as appropriate, an expected COR value is generated. Then, this modified forecast information is redesignated as the new observation, and the subsequent forecast is input for modification. Thus, the final terrain-modified track derived from these options is always tied to the input forecasts. That is, the position modifications are not allowed to build up. Although this is not appropriate for an operational forecast, it is ideal for testing terrain-induced intensity and translation speed modifications separately from the position modifications.

It was initially assumed that intensity and translation speed modifications would have to be tested separately. However, the results are identical when both modifications are tested because the minor speed modifications induced by terrain effects cause correspondingly minor changes in the time to/since storm crossing into the South China Sea. As indicated in Chapter 3, it is this time to/since seafall that determines how the intensity is modified, and these minor speed changes rarely changed the seafall time category with respect to the unmodified translation speed case. Thus, speed and intensity modifications can be calculated simultaneously.

The technique starts with one best track position and intensity as an observation, and uses the next as the forecast. After this forecast is modified as appropriate, the position modification is removed and it is redesignated as the next observation. Subsequent best track positions, intensities and translation speeds are used as further "forecasts" as the procedure iterates (see Fig. 12). It is critical to note that the intensity and translation speed estimates are the result only of continuous modification of the original observation values.

To test all possible combinations of the input, separate test runs were done on all storms in the data set in Table 2 using each best track data point within 360 n mi of Cubi Point as the initial observation. This was done with all best track data points that allowed at least a 6 and 12 h forecast to be generated within the 360 n mi radius. The final number of test runs generated by a given storm then depends on the speed and path of the storm through the area. For example, TY Nancy remained within 360 n mi of Cubi Point for only 24 h, or five best track points, while TY Cary provided 96 h of data, or 18 best track points.

Translation speed and intensity modifications were compared with the JTWC errors calculated for 24, 48 and 72 h forecast intervals for all storms. Only two of the 18 storms in the test sample (TY Phyllis and TY Cary) were in the Philippines area for 72 h or more. This constitutes too small a sample for meaningful statistics on 72 h forecast accuracy. Therefore, only the comparison with the JTWC 24 and 48 h error data are presented for magnitude and bias of the intensity and speed forecast errors. Only the JTWC errors for forecasts that matched the test forecast verification times were utilized, which ensures that this is a homogeneous test.

2. Intensity and Translation Speed Modification Results

For these results, intensity and speed values are the result of a series of terrain modifications on the original best track intensity and speed, given the best track positions. Ideally, the modifications should reflect the changes in intensity and translation speed as the storm approached and crossed the Philippines. The only exception is expected to be the first (6 h) forecast, which will be wrong anytime modifications occur, since the "forecast" using best track data is accurate.

The results of these tests are summarized in the graphs in Fig. 17 through Fig. 20. There are two graphs each for intensity and speed to illustrate the differences caused by the inclusion to the three tropical storms that were available (see Table 2) The studies that the intensity and speed rules were derived from excluded any storms not

reaching typhoon strength, so the "typhoon only" set of results theoretically should be more accurate. That this is not the case indicates (based on this limited sample) that tropical storms can be forecast using the algorithm.

Intensity forecast statistics in either case (Fig. 17 and Fig. 18) show program error magnitude increasing with forecast interval as expected with any forecast. As surmised above, there does appear to be more than expected error at the 6 h forecast. The intensity bias is monotonically increasing positive with increasing forecast interval, indicative of a systematic overforecasting of storm strength.

Error statistics derived from the corresponding JTWC 24 and 48 h forecast intensity error data are included in Fig. 17 and Fig. 18. JTWC intensity error magnitudes are 25% lower than the program forecast errors, although the standard deviation in both sets of statistics is high enough that this difference is not significant at the 95% confidence level based on the two sample difference test.

JTWC intensity error bias values for 24 and 48 h forecasts are positive in either case, with the 48 h forecast value being larger. This indicates that, as in program results, JTWC intensity forecasts tend to be uniformly too high. This supports the idea that some factor affecting intensity over the islands is being overlooked by the program rules since they do not correct this tendency.

Translation speed errors also show a general increase in magnitude with forecast interval (Fig. 19 and Fig. 20). The exception in either case is the apparent increase in 48 h forecast speed accuracy relative to the 42 h forecasts. However, this apparent improvement is not statistically significant at the 95% confidence level. The JTWC 24 and 48 h translation speed forecast errors are again 25% below the program errors for corresponding forecasts. Due to the high standard deviation on both JTWC and program error magnitudes, the difference between the two error sets is not statistically significant at the 95% confidence level.

The bias of the translation speed error statistics is more complex than for intensity errors. Forecast speeds are increasingly too high out to 18 h, decrease to nearly zero at 30 h, and continue to decrease until forecasts are significantly too slow for 48 h forecasts. JTWC speed error bias values are very nearly zero in either case at both forecast intervals, although the 48 h value is lower than the 24 h value, similar to what was noted for the program data. Thus, there is no systematic speed error in the JTWC forecasts as there is in the program. Clearly, the origin of this systematic error in the rules must be eliminated.

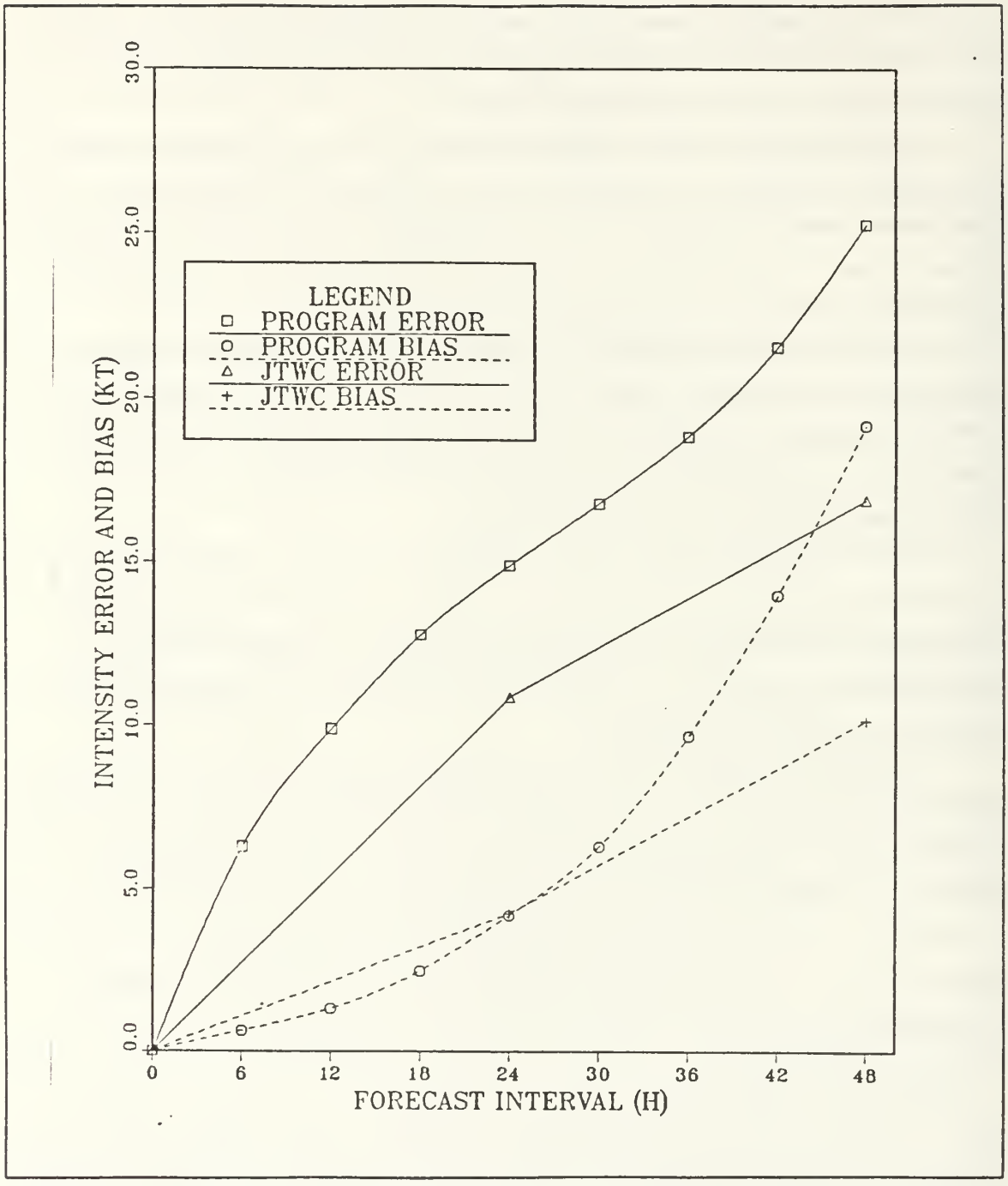

Fig. 17. Intensity forecast errors (TY and TS): Intensity error statistics as a function of forecast interval for the entire test set of storms compared to error figures calculated for a homogeneous set of JTWC forecasts.

A possible explanation for the intensity and speed biases might involve conditions that typically occur in the longer forecasts, say 30 h or more. The storm must

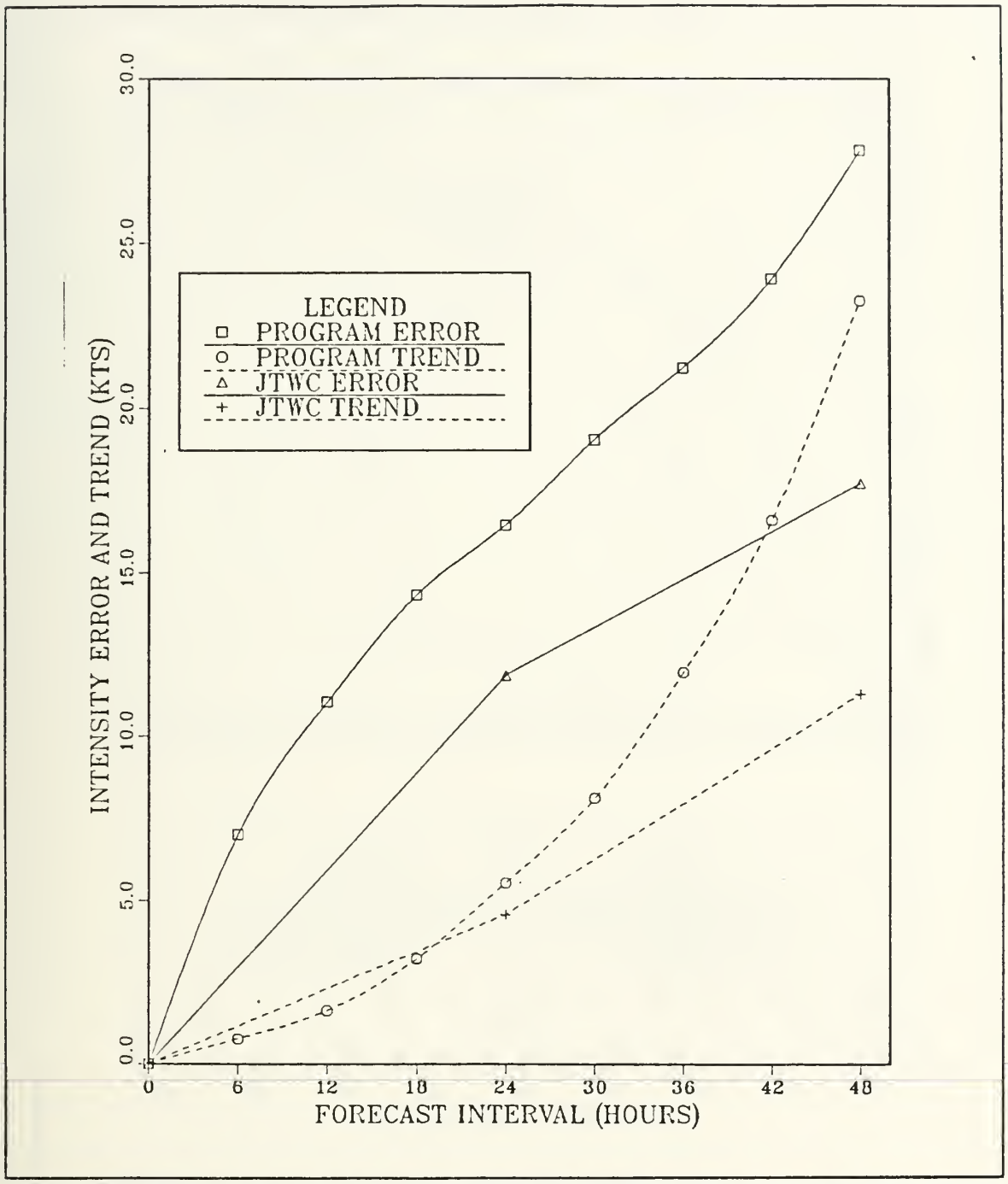

Fig. 18. Intensity forecast errors (TY only): As in Fig. 17, except using data only from those storms in the test set having a typhoon classification.

interact with the Philippine land mass at some point during these intervals. This is not necessarily the case for the shorter forecasts, however, which might include only over

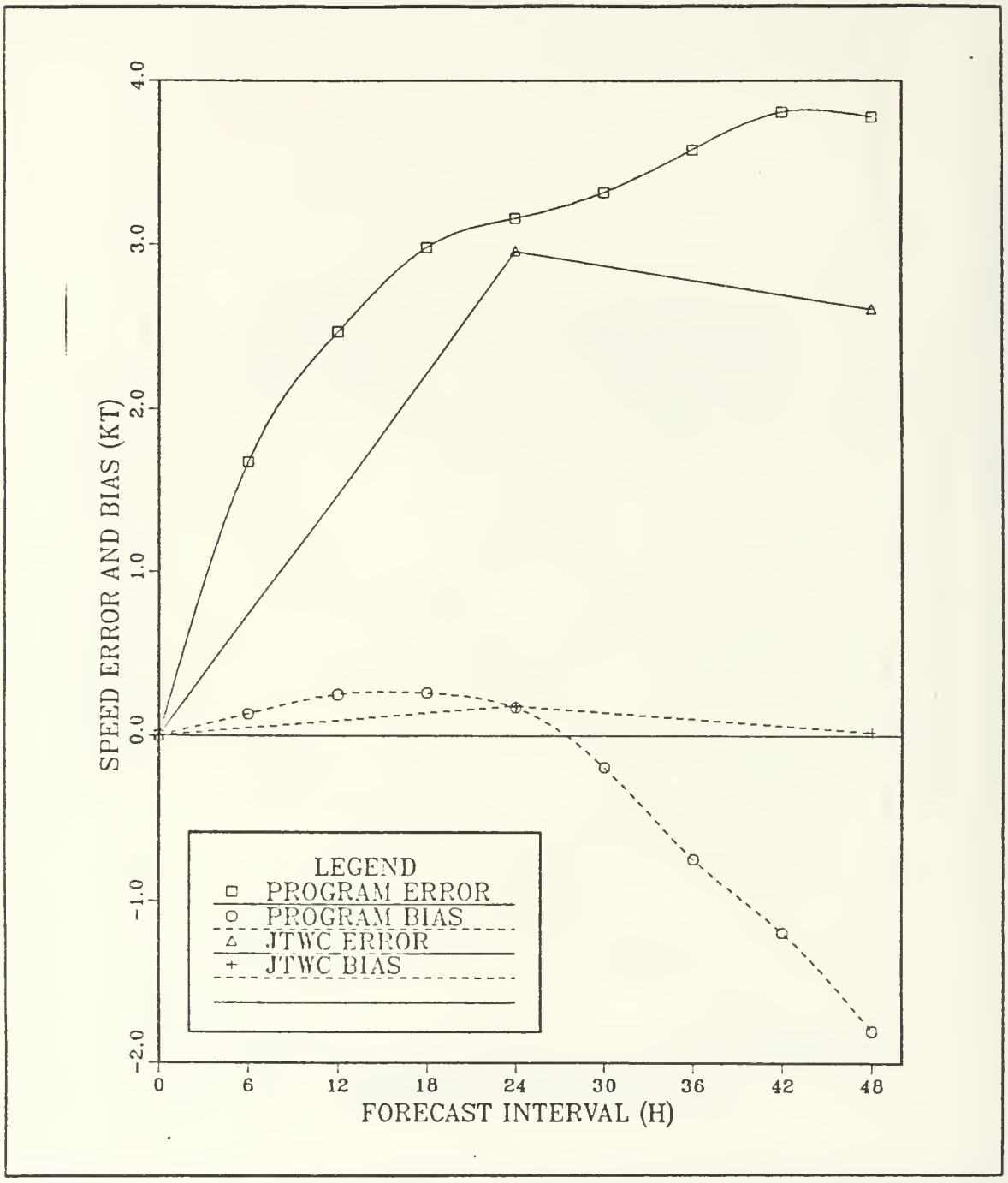

Fig. 19. Speed error statistics (TY and TS): Speed error statistics as a function of forecast interval for the entire test set of storms compared to error figures calculated for a homogeneous set of JTWC forecasts.

water conditions. It is possible that the rules used to modify the storm characteristics omit some factors that reduce storm intensity and increases translation speed as the

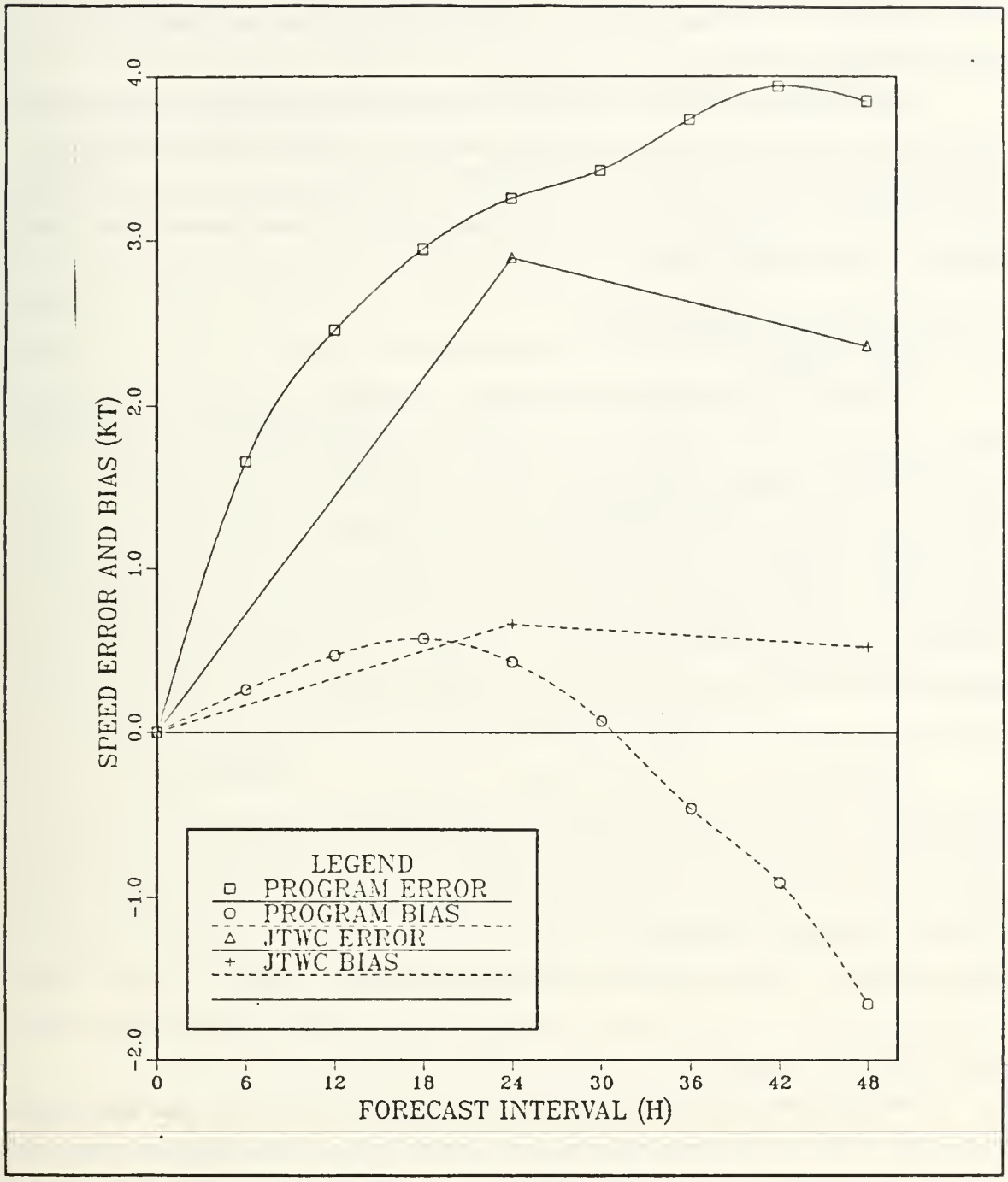

Fig. 20. Speed error statistics (TY only): As in Fig. 19, except using data only from those storms in the test set having a typhoon classification.

storm interacts with the islands. Thus, the error magnitudes and biases indicate that the storms are predicted generally to be more intense and slower moving than observed.

These uncertainties would be in addition to the normal errors that would build up as the forecast interval increases.

In light of the accuracy of the JTWC translation speed forecasts, it might appear best to remove program modifications in this area. It must be recalled though that the test input speed values were derived directly from the best track positions, and would not be available to the operational user of the program, who would have to get speeds from the available forecast products.

E. POSITION MODIFICATION

1. Test Procedure

This phase of the testing is to determine the overall accuracy of the terrain-modified position forecasts. While it is possible to force the algorithm to accept a correct, unmodified position while testing the translation speed modifications, it is not possible to generate position modifications without including translation speed modifications. This results from the assumption that a modified speed of advance over a forecast interval must result in a new forecast position. In particular, modifications to the forecast speed provide the along-track portion of the position modification. The position errors derived in this test procedure will need to be analyzed in conjunction with the translation speed error statistics generated in the test procedure described above.

Program options 2 and 5 are used for this phase of the testing (Fig. 13). For testing, option 2 requires sequential best track position and intensity data as an observation and as a "forecast", and modifies the latter. Option 5 then uses the course and distance from the first observation to the modified forecast position to extrapolate a new forecast from the previous one, and modifies this new forecast. Subsequent forecasts are generated in a similar manner using the two previous modified positions. This process generates a storm track that is connected by the input data to the best track only at the original observation.

As in the intensity and speed modification testing procedure, the position modification test is run on all storms using each best track positions that generate at least a 6 and 12 h forecast within the study area as an observation. Error statistics include north/south, east/west and total forecast error with respect to best track positions. Comparisons are made with a homogeneous set of JTWC 24 and 48 h forecasts.

2. Position Errors

Predicted total and component position error magnitudes are displayed in Fig. 21 for the case of typhoons only (no significant differences are found when tropical

storms are included). As expected, all of the error magnitudes increase steadily as the forecast interval increases. Notice that the total error is mostly due to north/south errors through 12 h, and mostly due to east/west errors for forecasts beyond 36 h.

Program north/south and east/west error bias figures are charted in Fig. 21. It is interesting to note that although there is an increasing north/south error magnitude, there is no systematic bias in this predominately cross-track direction. However, a large bias is found in the east/west, or along-track, direction, especially for forecast intervals greater than 24 h. This results from the slow speed forecasts, which cause position forecasts to be too far to the east for these westward moving storms. The speed bias for less than 24 h forecasts is negligible, so some other factor must be responsible for the east/west errors in this time range.

The corresponding JTWC 24 and 48 h total position errors are included in Fig. 21. At 24 h, JTWC total error is slightly greater than that of the program, while JTWC is more accurate at 48 h. Most of the gain in accuracy in using the program at 24 h comes from more accurate along-track prediction. As noted earlier, the program accuracy in along-track prediction suffers due to erroneous speed modifications beyond this time. Thus, the 48 h forecast advantage of JTWC is due to the inaccurate along-track forecasts at this time interval that the program produces.

3. COR Predictions

This phase of the testing most closely resembles operational use of the program, i.e. forecasts are generated solely on the basis of a current observation and one, albeit perfect, 6 h forecast. Therefore, the resulting COR values were examined to indicate the overall accuracy of the algorithm. While no hourly surface wind observation was over 35 kt, six cases occurred in which the peak winds recorded in conjunction with storm passage exceeded this value.

It is assumed that the highest COR predicted in any portion of the forecast represents the appropriate COR to set at the observation time. By comparing each of the best track observation times with the peak wind times recorded in the NOCF Cubi Point wind logs, the appropriate condition that should result from that observation can be determined. Comparisons of the predicted COR's with the appropriate wind settings are tabulated in Table 4.

A total of 47 of the 145 COR forecasts (32%) are correct. As indicated by the overprediction tendencies for wind estimation and storm intensity predictions noted in sections 2 and 3 above, the results indicate a strong overforecast bias for final COR

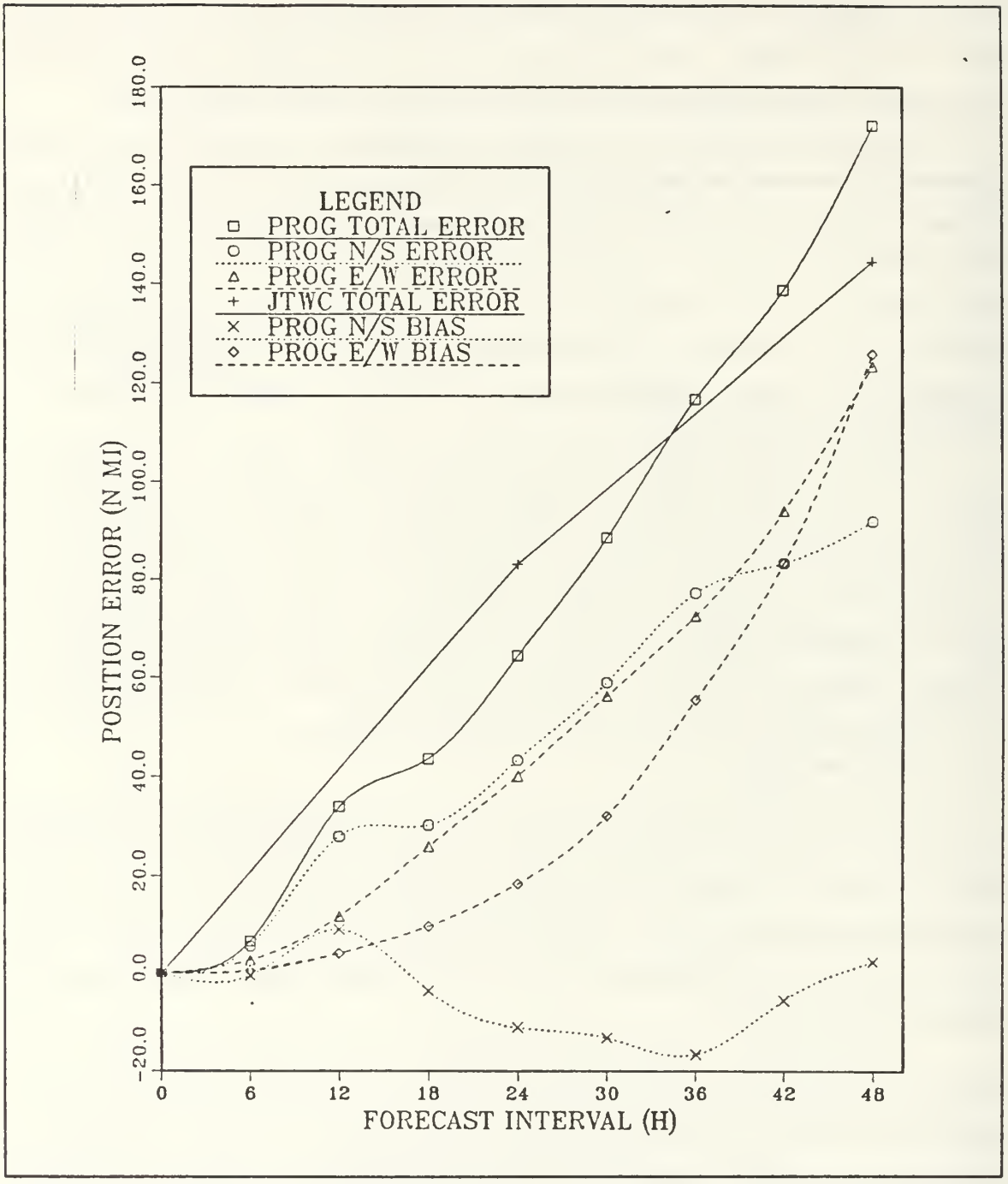

Fig. 21. **Position error statistics (TY only):** Program prediction and JTWC total forecast error, and program component position errors for all typhoons in the test set. Program bias curves are included.

values. This bias is not surprising, since as mentioned earlier, the maximum ratios were chosen to generate worst-case estimates for the winds at Cubi Point. The reasoning was

that it is better at this early stage in the development of the expert system to overforecast the COR in order to capture as many of the actual occurrences of 35 kt winds as possible. Comparing the number of predictions of COR 3 or higher with the total number of cases where COR 3 or higher is appropriate from Table 4 results in a 95% capture rate. This means that in 95% of the cases in which 35 kt or greater peak winds were observed, the algorithm also indicated 35 kt or greater winds in conjunction with storm passage. The predicted time of arrival of these winds may be inaccurate, contributing to the low overall COR accuracy percentage, but the magnitude of the winds is correct.

Table 4. PREDICTED VS. APPROPRIATE COR SETTINGS: Comparison of appropriate COR settings, based on peak winds logged in conjunction with tropical cyclone passage, and program predictions. Total cases overall and for each appropriate COR are indicated along the bottom of the table.

		Actual COR				
		1	2	3	4	
Predicted COR	1	13	11	6	54	
	2	1	2	4	16	
	3	0	0	0	4	
	4	2	0	0	32	
		16	13	10	106	Total = 145

The price to be paid for achieving this high capture ratio is in the percentage of false alarms, when winds are erroneously predicted in excess of 35 kt. Due to the choice of the maximum gust ratios for wind estimation, 70% of the cases that did not produce winds over 35 kt were falsely predicted to merit COR 3 or higher. Thus while the vast majority of the cases in which potentially destructive winds occurred are detected by the algorithm, much time and resources would be wasted protecting Navy assets from nonexistent destructive winds if the algorithm were the only input into the setting of COR's.

F. CLIMATOLOGICAL TRACKS

Options 6 and 8, and 7 and 8 will not be explicitly tested for this study. These options allow forecasts of local Cubi Point winds and COR assuming that the storm follows either the closest, or a user chosen, climatological track over the Philippines,

respectively. It was determined that all sources of error involved in using these options that could be tested had been examined in the test procedures described above.

There are four possible sources of error that affect the accuracy of the climatological track options. These error sources are errors in determining the climatological tracks, errors in estimating storm travel along these tracks, errors in estimating storm intensity and errors in computing the resulting local winds at Cubi Point

As mentioned earlier, these climatological best tracks were taken from a study by Sikora (1976). The details on track derivation and estimates of their accuracy must be based on information from that paper. It is beyond the scope of this study to reanalyze the tracks, so the working assumption is that they are accurate.

With this assumption, the only three potential sources of error remaining (intensity, speed and wind estimation) are identical to the potential error sources involved in the other options. These aspects of the program will be thoroughly tested since the application of the modifications in the climatological track options is identical to that of the other options. The only difference is in how the ground track of each forecast is determined, and for these options they are assumed to be correct. Therefore, the error statistics collected through the other options apply equally to the climatological track options.

VI. SUMMARY AND CONCLUSIONS

A. SUMMARY OF RESULTS

A summary of the error magnitudes of both the program results and of JTWC forecasts is included in Table 5. These results shows that the prototype program can, given perfect input, produce results that are comparable with those of the current official position and translation speed forecasts. Any errors in the working position and intensity data available to the forecaster should increase the program errors, although the actual effect has not been studied.

Table 5. ERROR MAGNITUDE SUMMARY (TY AND TS): Summary of major error statistics for program tests. JTWC forecast errors for a homogeneous test set of storms are listed when applicable. Official Cubi Point COR settings were not available.

Error type	Program	JTWC
24 h speed (kt)	3.2	3.0
48 h speed (kt)	3.8	2.6
24 h intensity (kt)	14.9	10.8
48 h intensity (kt)	25.3	16.9
24 h N/S (n mi)	46.0	N/A
24 h E/W (n mi)	41.0	N/A
48 h N/S (n mi)	91.1	N/A
48 h E/W (n mi)	136.3	N/A
24 h total (n mi)	67.7	84.8
48 h total (n mi)	181.3	155.3
Local gust forecasts (kt)	20.6	N/A
COR accuracy (%)	32	N/A
Capture rate (winds > 35 kt) (%)	95	N/A
False alarm rate (%)	70	N/A

The program returns local wind forecasts given actual storm position and intensity that are too high on average. This error is due to the comparison of program estimates of maximum expected gusts with hourly recorded values that are most like mean gusts. Another source of error is the high standard deviation in the values of the gust ratios

calculated by Jarrell and Englebretson (1982). The introduction of rules based upon an expanded study of the ratio between storm strength and local winds can reduce these errors if the standard deviation in the study can be reduced below the Jarrell and Englebretson (1982) values.

The program predicts erroneously low storm speeds beyond 24 h that are reflected in 48 h position forecasts too far to the east. This error seems to result from inappropriate application of the speed rules based on Sikora (1976). Either the storm variability in the region is too high to allow application of the mean speed values, or the sample used to generate the mean profiles is too small. These errors may be corrected by further study. If the error in the program results remain with an updated rule base, it is likely that empirical rules will be required to ensure proper application of the speed rules.

The north/south correction added in the algorithm to forecast storm position is effective, in the sense that no north/south bias in the modified positions is noted. This indicates either appropriate application of the correction, or that the correction is simply too small to be significant compared to the north/south component of the speed correction.

Program intensity forecasts are in general higher than the post-analysis intensities. This is also the case to a lesser degree for corresponding JTWC intensity forecasts. This similarity is not surprising since JTWC forecasters use the Sikora (1976) study when making forecasts for storms in the Philippine region (personal communication with LT Mark Gunzelman). The improvement noted by JTWC intensity forecasts over the program results then is due to the human discretion and experience involved in JTWC forecasts that is not yet included in the expert system.

As explained for the speed errors earlier, the source of the errors in the program intensity forecasts may be the result of an erroneous application of, or an underestimate of, the errors of the Sikora (1976) intensity profiles. A wider-based study and/or empirical rules to allow better application of the modifications are required.

The program position forecast error at 24 h actually exhibits less error than the JTWC 24 h forecasts. Program accuracy diminishes out to 48 h where JTWC forecasts are more accurate. The main source of the program error beyond 24 h is the erroneously low translation speed estimates, which cause the position forecasts to exhibit an easterly bias.

Finally, when all modification factors are combined to arrive at COR recommendations, the results of this limited study are only 32% accurate. The vast majority of the

erroneous recommendations indicate an overforecast bias, since the maximum expected gust values were used for the predictions. This bias was included deliberately to ensure that the program would not underforecast the COR in cases when potentially destructive winds actually occur. The capture rate is thus a very high 95% for those cases where 35 kt winds actually were observed. However, there is a correspondingly high false alarm rate of 70% when cases not justifying COR above 4 were improperly forecast by the algorithm.

B. AREAS NEEDING IMPROVEMENT

The prototype program is more accurate for forecasting position than storm strength or translation speed, even though speed errors directly contribute to position errors. Position forecast accuracy and the accuracy of the COR estimates lead to the conclusion that even with these errors, the prototype algorithm is useful.

Intensity and translation speed forecast errors seem to have no easy explanation. If the algorithm is correct, then the results of the previous studies that formed the rule base do not appear to apply to this sample. This may indicate that more must be learned about terrain influences on storm parameters. An interim solution to the intensity overprediction problem could be to include a correction factor to reduce program generated estimates, since the trend is always toward overprediction. However, this would lead to more errors due to underforecasting the COR values than occurs presently.

A similar solution is possible to correct the uniform overprediction of the Cubi Pt. local winds. However, this option would also enlarge the bias towards underprediction of COR's and is based solely on statistical rather than physical reasoning. This technique of incorporating a correction factor would not be useful in correcting the translation speed errors since there is no distinct trend over all forecast intervals. For all of these reasons, it is preferable to await further study into the dynamics of terrain-tropical storm interactions before improvements are attempted in these areas.

Another possible interim solution to the intensity forecast problems is to eliminate this section altogether, and use only the JTWC forecast intensities as issued, since they are 25% better than the modified estimates. Alternately, some combination of the two, such as applying modifications only in certain cases, could prove useful. More study into the physics would help decide on the best method to employ and how best to implement it.

The data set was adequate for the test procedure. Although storms that cause local winds at Cubi Point over 35 kt are fairly rare due to the excellent protection offered by the local terrain (Douglass, 1975), six such cases were included in the data set. One improvement might be to deal with a larger sample of these more dangerous storms to get a better idea of how the algorithm handles these cases which are more significant in terms of human activity than those with lower winds.

To reduce the false alarm rates in those cases where no destructive winds occurred, gust ratios other than the maximum should be studied. An optimum balance must exist between a high capture rate and a low false alarm rate that will result in predictions of COR that provide an acceptable level of protection without an unacceptable level of unnecessary expense. The average gust ratio, for example, may provide this balance.

Finally, one fundamental weakness in the prototype program can be improved upon. As originally envisioned, the expert system was to include rules obtained from expert forecasters, such as empirical rules derived operationally by Typhoon Duty Officers. In the prototype, the main emphasis was on generating a working product based on available rules from the literature. Therefore, the rules are more numerical and statistical than empirical. Although this does not invalidate the results or the approach, it does mean that maximum advantage has not been taken of the traits and techniques peculiar to PROLOG. Potential improvements are available through including any rules the experts have generated.

C. CONCEPT VALIDATION

This project was designed to construct a prototype expert system to demonstrate the feasibility of an expert system in hurricane forecasting. It was not expected that this prototype would produce more accurate forecasts than are currently available. In this light, the results are encouraging, and the prototype is a success.

Given the limited experience available at the start of this project (both the PROLOG language and the theory of expert systems had to be learned), the success achieved in just nine months by one individual is very encouraging. Considering the limited number of man-hours expended to get these useful and accurate results from the program, the results possible by using more man-hours and more expertise could be tremendous.

A skilled expert system engineer would be very useful to reduce wasted effort and enhance efficient rule generation. Even in the absence of such an engineer, a skilled meteorologist working with as many typhoon experts as possible could add significantly to the program's rule base, and thus its accuracy and usefulness.

In the case of terrain modifications of typhoons, much can yet be gained by including the input of experts. When the expert system concept is expanded to other facets of typhoon behavior, it may not be as easy to gather rules either from published studies or from experts. It would be useful in the case of the former to have the research done specifically with an eye towards eventual use of the results in an expert system, which might ease their translation into PROLOG.

Based on an overall examination of the results of this project, the concept appears quite valid -- expert systems have the potential to be applied successfully to the problem of typhoon forecasting. The limited time and expertise brought to bear on the problem nonetheless were able to achieve useful and reasonable accurate results. Furthermore, plenty of room for even better results appears available by bringing in more expert knowledge and experience to enhance the rule base. The avenue of using expert systems in hurricane forecasting shows great promise and should be explored further.

APPENDIX A. TURBO PROLOG PRIMER

A. INTRODUCTION

The first version of PROLOG (**Pro**gramming in **Log**ic) was created in the early 1970's by Alain Colmerauer at the University of Marseilles. PROLOG is increasingly popular, and is becoming accepted as the standard for small-scale artificial intelligence/expert system applications. While several versions of PROLOG are available, they all have similar structure.

The term programming really does not apply to PROLOG products in the conventional sense of FORTRAN, BASIC or other languages commonly used in numerical applications are known as procedural languages. Procedural languages require the programmer to create the algorithm, or procedure, that is to be followed exactly when the program is executed. Such a procedure-oriented program will always follow the same path through the set of instructions until the conclusion is reached. Each calculation or branch in the program's flow must be explicitly prescribed by the programmer in the code.

In contrast, PROLOG is a declarative language, and the logic does not require an exact sequence of steps to be followed to arrive at a conclusion or result. Rather, the program (the term, though inaccurate, is still used to describe a discrete set of PROLOG code) consists of a set of facts and a set of rules related to the problem under consideration. In response to a query, a PROLOG program accesses only those facts and rules necessary to determine all possible solutions. No instructions have to be included to specify how to sort through the facts or rules. The structure of PROLOG allows different paths to be taken if necessary with each program run.

B. PROLOG FACTS

Facts are written in so-called predicate form, and represent objects and their relationships. These objects can be numbers, people, things, words or whatever is needed for the problem. A typical example of predicate structure is *owns(lauren, bicycle)*[2], where the predicate "owns" describes the relationship between its two arguments

---

2 Throughout this appendix, Turbo PROLOG syntax will be observed. Predicates and their constant arguments will be represented by names beginning in lower case letters, and variable names will begin with upper case letters.

"lauren" and "bicycle". PROLOG facts can represent abstract ideas such as ownership, or more concrete relationships such as *weight(bob, 185)* for "Bob weighs 185 lbs.", or *father(julie, george)*, for "Julie's father is George". Facts can have any number of objects as their arguments, depending upon what is to be represented. For example, one method to represent the fact that Johnny's parents are Fred and Sybill is *parents(johnny, fred, sybill)*.

In PROLOG facts, the specifics of the relationships between various objects is never explicitly defined. It is enough to simply assert that a relationship (e.g. father, owns, weight) does exist, whatever the form or implications of that relationship might be. The names given to the objects or the relationships are really irrelevant and (with a few exceptions noted later) have no intrinsic meanings in PROLOG. The significance of the objects' names is in their location in a predicate. For instance, *father(a, b)* and *father(c, d)* mean that the relationship that is implied by "father" holds for "a" and "b" also applies for "c" and "d". Although no rule exists that states that the "father" relationship must imply that one object is the male parent of the other, common convention and good programming practice dictate that object and predicate names should be chosen to make the program read as much like standard English as possible.

To illustrate how a set of facts is used in PROLOG to respond to a query, consider the program in Fig. 22 consisting only of a set of facts about ownership. In Turbo PROLOG, the "predicates" paragraph consists of statements of all predicates used in the program with designation of their argument types. The "clauses" paragraph lists the actual facts and rules (discussed later) that comprise the program.

If the query *?-owns(sally, dog)* is input, the PROLOG program will execute a search the data facts to see if the query can be proven true. Since there is a fact in the fact base which exactly matches the query, the program outputs *TRUE*.

If the query *?-owns(john, X)* is entered, the response would be a list of objects which John owns, specifically

X = ball

X = bike

2 solutions. This output contains all values for the query variable X found after a search of the data base that make the query a true statement. In Turbo PROLOG, the entire data base is searched and all values that satisfy the query are outputted, while other versions of PROLOG only return the first such value.

```
predicates
owns(symbol, symbol)
clauses
owns(john, ball).
owns(jack, ball).
owns(john, bike).
owns(sally, dog).
```

Fig. 22. Example PROLOG program 1.: Simple PROLOG program describing ownership of several objects from which inferences about ownership can be made using queries.

The query could also be entered without any constants named in the argument list, such as *?-owns(X, Y).* The result will be all pairs of X and Y that satisfy the query:

X = john, Y = ball

X = jack, Y = ball

X = john, Y = bike

X = sally, Y = dog

4 solutions.

C. PROLOG RULES AND BACKTRACKING

Logic is the tool that allows generation or inference of new facts from the knowledge base of facts included in the PROLOG program. The rule of logic most often used in PROLOG is called *modus ponens*, which simply says that if fact A implies fact B, and fact A is true, then fact B must be true as well. "If Sue is a mother, and if mothers have a child, then Sue has a child" is an example of the application of *modus ponens.*

Modus ponens is incorporated into all PROLOG rule statements included in programs. A rule corresponding to the statement about Sue could be *has_child(sue):-mother(sue), has_child(mother)*, where the symbol ":-" roughly translates as "if", and the comma means "and". Notice that in PROLOG structure, the conclusion is on the left side of a rule statement, and the conditions are on the right side.

PROLOG relies on a method called backtracking to reach a conclusion when the query involves rules. The first step in backtracking is to find the rule that has to be true or false to answer the query. PROLOG then tries to satisfy the conditions of this rule, which may involve further rules. Eventually, this backward tracking through the layers of rules will lead to the known facts, which allows an evaluation of the rules back toward

the original rule. Only when all of the conditions of the original rule applying to the query are found to be true is the answer *TRUE* or are the appropriate values of any query variables printed out. If any of the conditions can not be satisfied, the response to the query is *FALSE*.

Consider the program in Fig. 23 (after Clocksin and Mellish, 1984) as an example to illustrate the backtracking process. In response to the query *?-uncle(elaine, Uncle)*, the program will first check the fact base to see if information about Elain's uncle is known. Not finding any facts that match the query, the program pointer shifts to the first rule it can find about uncles in general. In this case, Elaine ("A") has an uncle "B" if she ("A") has parents "X" and "Y", and if "Y" has a brother "B". From the "parents" data, "Y" is equal to Alice. With this "parents" condition satisfied, the program tries to satisfy the second condition, which becomes *brother(alice, B)*, where "B" is the answer for "Uncle" desired in the query. From the relative positions of the arguments in the "brother" condition, "C" becomes "alice", and "D" becomes "Uncle". Because there is no fact or rule to allow *parents(alice, X, Y)* to be satisfied, the "brother" rule fails. This failure will cause a transfer back to the first "uncle" rule, which thus fails.

The program pointer then moves to the second rule for uncle. Since the first condition can be satisfied just as before, the program moves on to try to satisfy the second condition, which has become *brother(bob, Uncle)*. Now, a fact exists that satisfies the first condition in the "brother" rule, so that "X" and "Y" become Bob's parents Fred and Barbara, respectively. The remaining three "brother" conditions are examined to find someone else whose parents are Fred and Barbara ("X" and "Y"), who is not the same as Bob, and who is a male. From the data, it is clear that Joe satisfies all these conditions, so Uncle is set to "joe". With the "brother" rule satisfied, all the conditions for the "uncle" rule are also satisfied, and the answer printed is

Uncle = joe

1 solution.

Another use of the same program can illustrate the capability in PROLOG to find all possible solutions to a query involving rules and facts. A question can be phrased with variables only, as in *brother(A, B)*, which asks the program to find anyone "A" who has a brother "B". Since none of the facts directly answers the question, the "brother" rule must be checked. Since no "uncle" information is required, these rules are simply ignored for this problem. Since rules and facts are always examined from top to bottom,

```
predicates
male(symbol)
female(symbol)
parents(symbol, symbol, symbol)
uncle(symbol, symbol)
brother(symbol, symbol)
clauses
male(jim).
male(bob).
male(fred).
male(joe).
female(alice).
female(elaine).
female(barbara).
parents(elaine, bob, alice).
parents(jim, bob, alice).
parents(bob, fred, barbara).
parents(joe, fred, barbara).
uncle(A, B):- parents(A, X, Y), brother(Y, B).
uncle(A, B):- parents(A, X, Y), brother(X, B).
brother(C, D):-
   parents(C, X, Y), parents(D, X, Y), not(C= D), male(D).
```

Fig. 23. Example PROLOG program 2.: Simple PROLOG program defining a set of family relationships. Suffucient facts about specific family members to make fairly complex inferences about these relationships using queries.

the first set that satisfies the conditions for "brother" are Elaine and Jim. However, all other possible solutions are sought in Turbo PROLOG, which results in

A = elaine, B = jim

A = bob, B = joe

A = joe, B = bob

3 solutions.

In PROLOG, the last two solutions are not the same, since they mean that Joe has a brother named Bob, and Bob has a brother named Joe, and neither implies the other. The program must be modified if these responses are considered to be redundant.

A distinctive feature of Turbo PROLOG is that it will display all possible solutions to a query, unless otherwise specified. Other versions of PROLOG require a carriage return after output of the first response for each additional answer. In all versions of PROLOG, the query can be input by the user each time the program is run, as in the above examples, or it can be an integral part of the program, as a "goal" that the program tries to satisfy automatically.

Another unique feature of Turbo PROLOG is that variable types must be explicitly declared, as in FORTRAN. Because the variables from the program above represent people, not real or integer numbers or character strings, they were declared as "symbols". Although this requirement can cause confusion in the programming and debugging stages, especially in a large program with many variables and predicates, it is necessary for the compiler in the Turbo PROLOG package to run efficiently.

Turbo PROLOG has many other significant features, most of which are shared with other versions of PROLOG. A large set of built-in predicates are provided, including some mathematical functions such as sin(x), cos(x), tan(x), exp(x), ln(x) and sqrt(x), as well as the standard mathematical operators and comparisons (+, -, *, /, >, <, =, etc.). Turbo PROLOG also has a unique set of commands that allows for various colored outputs, as well as graphics and window commands.

This appendix serves as only a brief introduction to the structure and use of PROLOG. It is a flexible but very complex language, and it is very different from the more familiar numerical languages. Clocksin and Mellish (1984) provide more information on other aspects of this language, including the powerful tools of PROLOG recursion and list processing techniques.

APPENDIX B. TROPICAL CYCLONE CONDITION FORECASTING PROGRAM

```
/*  TTTTT  H   H  EEEEE  SSSSS   III   SSSSS      QQQQQ   */
/*    T    H   H  E      S        I    S          Q   Q   */
/*    T    HHHHH  EEE    SSSSS    I    SSSSS      Q Q Q   */
/*    T    H   H  E          S    I        S  xx  Q  QQ   */
/*    T    H   H  EEEEE  SSSSS   III   SSSSS  xx  QQQQQ   */

/* THESISQ.PRO */
/* Written by LT. Bruce M. Hagaman on 13 July 1988  */

code = 4000

domains
 file=resultfile
 posit=real*
 plist=posit*
 slist=string*

database
  obs(real,real,real,real,integer,real,real)
  forecast(real,real,real,real,integer,real)
  seafall(real)
  mod_forecast(real,real,real,real,integer,real,real)
  base_time(real,real,integer)
  obs_time(real,real)
  forecast_int(real)
  response(string)
  p(real)
  posit_list(plist)
  cond_list(slist)
  gust_list(posit)

predicates
 start
 input_obs
 cpa(real,real,real)
 input_forecast
 modify_forecast
 strength(real,real,real,real,real)
 posit(real,real,real,real,real,real,real,real,real,real,real)
 speed(real,real,real,real,real,real)
 seg_dist_ts(posit,real)
 seg_dist_tc(posit,real)
 result(string,real,real)
 condition(string)
 wind(real)
 time_diff(real,real,real,real,real)
 radius_to_cubi(real,real,real)
 ts(real,posit,real,real)
 tc(real,posit,real,real)
```

```
 closest_seg(plist,real,real,real)
 dist(real,real,real)
 range(real,real,real,real,real)
 ts_factor(real)
 tc_factor(real)
 do(integer)
 xtrack(real,real,real,real)
 menu(real)
 ave_strength(real,real,real)
 heading(real,real,real,real,real,real)
 xtrap(real,real,real,real,real,real)
 newtime(real,real,integer,real,real,integer,real)
 path(real,real,real,real)
 path_point(posit,real)
 closest_path(plist,real,real,real,real)
 do_more(real)
 append(plist,posit,plist)
 append(slist,string,slist)
 append(posit,real,posit)
 write_output(plist,posit,slist)
 store(posit,string)
 write_list(posit,real)
 store_output(string)
 instruct

goal
 start, instruct, input_obs, menu(0).

clauses
 start:-makewindow(1,7,15,"Tropical Cyclone Forecaster",3,0,22,80),
    assertz(forecast(1,1,1,1,1,1)), assertz(mod_forecast(2,2,2,2,2,2,2)),
    assertz(forecast_int(0)),assertz(p(0)).

 instruct:-
    write("This program will ask for tropical cyclone observation "),nl,
    write("data.  Then, you will be given a choice of several "),nl,
    write("options.  These options will generate Conditions of "),nl,
    write("Readiness forecasts based on this observation and a "),nl,
    write("forecast/series of forecasts.  Terrain modifications "),nl,
    write("may be included or not, or the storm can be tracked "),nl,
    write("along a climatological track across the Philippines.  "),nl,
    write("Single forecast results or iterations can be chosen.  "),nl,
    write("Upon completion of the run based on one observation, the "),
    nl,
    write("highest COR resulting at any step indicates the recommended"),
    nl,
    write("COR appropriate to be set at observation time."),nl,
    write("To continue with this run, hit the space bar and respond "),
    nl,
    write("to the prompts on the screen."), readchar(_).


/* these facts are from the Jarrell and Englebretson study  */
 ts(1,[14.8,120.3],1.045,0.364).
 ts(2,[16.0,121.0],0.857,0.338).
 ts(3,[14.8,121.8],0 750,0.184).
```

```
ts(4,[13.6,121.0],0.532,0.254).
ts(5,[13.6,119.6],0.789,0.257).
ts(6,[14.8,118.8],0.800,0.225).
ts(7,[16.0,119.6],0.914,0.353).
ts(8,[17.1,121.1],0.581,0.303).
ts(9,[16.2,122.4],0.480,0.195).
ts(10,[14.8,122.8],0.360,0.146).
ts(11,[13.4,122.4],0.563,0.182).
ts(12,[12.5,121.1],0.600,0.189).
ts(13,[12.5,119.5],0.500,0.193).
ts(14,[13.4,118.2],0.550,0.194).
ts(15,[14.8,117.8],0.333,0.184).
ts(16,[16.2,118.2],0.560,0.216).
ts(17,[17.1,119.5],0.675,0.342).
ts(18,[18.2,121.1],0.900,0.301).
ts(19,[17.5,122.5],0.458,0.162).
ts(20,[16.3,123.5],0.489,0.184).
ts(21,[14.8,123.9],0.567,0.196).
ts(22,[13.3,123.5],0.300,0.116).
ts(23,[12.1,122.5],1.000,0.266).
ts(24,[11.4,121.1],0.400,0.185).
ts(25,[11.4,119.5],0.500,0.270).
ts(26,[12.1,118.1],0.385,0.214).
ts(27,[13.3,117.1],0.900,0.213).
ts(28,[14.8,116.7],0.433,0.186).
ts(29,[16.3,117.1],0.380,0.187).
ts(30,[17.5,118.1],0.640,0.263).
ts(31,[18.2,119.5],0.667,0.269).
ts(32,[19.2,121.1],0.600,0.234).
ts(33,[18.7,122.6],0.526,0.181).
ts(34,[17.7,123.8],0.467,0.190).
ts(35,[16.3,124.7],0.560,0.136).
ts(36,[14.8,124.9],0.667,0.151).
ts(37,[13.3,124.7],0.550,0.142).
ts(38,[11.9,123.8],0.833,0.194).
ts(39,[10.9,122.6],0.476,0.186).
ts(40,[10.4,121.1],0.483,0.232).
ts(41,[10.4,119.5],0.333,0.185).
ts(42,[10.9,118.0],0.480,0.237).
ts(43,[11.9,116.8],0.480,0.211).
ts(44,[13.3,115.9],0.450,0.195).
ts(45,[14.8,115.7],0.467,0.187).
ts(46,[16.3,115.9],1.111,0.271).
ts(47,[17.7,116.8],0.524,0.192).
ts(48,[18.7,118.0],0.720,0.213).
ts(49,[19.2,119.5],0.658,0.224).
ts(50,[20.2,121.1],0.500,0.223).
ts(51,[19.8,122.7],0.500,0.216).
ts(52,[18.9,124.0],0.500,0.256).
ts(53,[17.8,125.1],0.800,0.187).
ts(54,[16.3,125.7],0.500,0.144).
ts(55,[14.8,126.0],0.786,0.215).
ts(56,[13.3,125.7],0.417,0.149).
ts(57,[11.8,125.1],0.480,0.231).
ts(58,[10.7,124.0],0.342,0.176).
ts(59,[9.8,122.7],0.320,0.168).
```

```
ts(60,[9.4,121.1],0.560,0.201).
ts(61,[9.4,119.5],0.560,0.296).
ts(62,[9.8,117.9],0.203,0.146).
ts(63,[10.7,116.6],0.480,0.249).
ts(64,[11.8,115.5],0.081,0.081).
ts(65,[13.3,114.9],0.350,0.172).
ts(66,[14.8,114.6],0.519,0.164).
ts(67,[16.3,114.9],0.600,0.253).
ts(68,[17.8,115.5],0.800,0.223).
ts(69,[18.9,116.6],1.120,0.253).
ts(70,[19.8,117.9],0.667,0.232).
ts(71,[20.2,119.5],0.438,0.237).

tc(1,[14.8,120.3],0.494,0.265).
tc(2,[16.0,121.0],0.448,0.167).
tc(3,[14.8,121.8],0.521,0.262).
tc(4,[13.6,121.0],0.308,0.180).
tc(5,[13.6,119.6],0.369,0.243).
tc(6,[14.8,118.8],0.338,0.200).
tc(7,[16.0,119.6],0.353,0.249).
tc(8,[17.1,121.1],0.314,0.149).
tc(9,[16.2,122.4],0.261,0.136).
tc(10,[14.8,122.8],0.267,0.125).
tc(11,[13.4,122.4],0.241,0.138).
tc(12,[12.5,121.1],0.194,0.109).
tc(13,[12.5,119.5],0.269,0.123).
tc(14,[13.4,118.2],0.271,0.143).
tc(15,[14.8,117.8],0.200,0.110).
tc(16,[16.2,118.2],0.185,0.102).
tc(17,[17.1,119.5],0.400,0.152).
tc(18,[18.2,121.1],0.313,0.156).
tc(19,[17.5,122.5],0.330,0.132).
tc(20,[16.3,123.5],0.277,0.120).
tc(21,[14.8,123.9],0.161,0.067).
tc(22,[13.3,123.5],0.253,0.103).
tc(23,[12.1,122.5],0.192,0.091).
tc(24,[11.4,121.1],0.076,0.076).
tc(25,[11.4,119.5],0.200,0.142).
tc(26,[12.1,118.1],0.165,0.091).
tc(27,[13.3,117.1],0.152,0.109).
tc(28,[14.8,116.7],0.165,0.100).
tc(29,[16.3,117.1],0.242,0.068).
tc(30,[17.5,118.1],0.359,0.124).
tc(31,[18.2,119.5],0.301,0.178).
tc(32,[19.2,121.1],0.341,0.148).
tc(33,[18.7,122.6],0.413,0.116).
tc(34,[17.7,123.8],0.275,0.080).
tc(35,[16.3,124.7],0.198,0.075).
tc(36,[14.8,124.9],0.169,0.073).
tc(37,[13.3,124.7],0.210,0.056).
tc(38,[11.9,123.8],0.175,0.084).
tc(39,[10.9,122.6],0.200,0.177).
tc(40,[10.4,121.1],0.154,0.137).
tc(41,[10.4,119.5],0.250,0.173).
tc(42,[10.9,118.0],0.157,0.113).
tc(43,[11.9,116.8],0.100,0.076).
```

```
tc(44,[13.3,115.9],0.185,0.081).
tc(45,[14.8,115.7],0.253,0.118).
tc(46,[16.3,115.9],0.453,0.141).
tc(47,[17.7,116.8],0.282,0.092).
tc(48,[18.7,118.0],0.373,0.155).
tc(49,[19.2,119.5],0.329,0.180).
tc(50,[20.2,121.1],0.254,0.153).
tc(51,[19.8,122.7],0.195,0.098).
tc(52,[18.9,124.0],0.261,0.099).
tc(53,[17.8,125.1],0.215,0.067).
tc(54,[16.3,125.7],0.198,0.076).
tc(55,[14.8,126.0],0.120,0.054).
tc(56,[13.3,125.7],0.123,0.051).
tc(57,[11.8,125.1],0.100,0.056).
tc(58,[10.7,124.0],0.192,0.095).
tc(59,[9.8,122.7],0.185,0.120).   /*RATIOS ESTIMATED*/
tc(60,[9.4,121.1],0.175,0.140).   /*RATIOS ESTIMATED*/
tc(61,[9.4,119.5],0.169,0.162).
tc(62,[9.8,117.9],0.092,0.068).
tc(63,[10.7,116.6],0.108,0.088).
tc(64,[11.8,115.5],0.200,0.093).
tc(65,[13.3,114.9],0.099,0.059).
tc(66,[14.8,114.6],0.126,0.076).
tc(67,[16.3,114.9],0.261,0.097).
tc(68,[17.8,115.5],0.115,0.076).
tc(69,[18.9,116.6],0.215,0.098).
tc(70,[19.8,117.9],0.194,0.110).
tc(71,[20.2,119.5],0.338,0.169).


/* the input_obs predicate serves as the input point for  */
/* the storm observation data.                            */
 input_obs:- write("Current observed storm latitude  - "), readreal(Olat),
    write("Current observed storm longitude - "), readreal(Olong),
    write("Observation time - "), readreal(Otime),
    write("Observation Julian Day - "), readreal(Oday),
    assertz(obs_time(Otime,Oday)),
    write("Observation year - "), readint(Oyear),
    assertz(base_time(Otime,Oday,Oyear)),
    write("Current max surface winds."), readreal(Oint),
    write("Input current average storm SOA."), readreal(Ospeed),
    write("Input current maximum gusts at Cubi."), readreal(Ogust),
    write("Input current TS or Typhoon condition"),nl,
    write("   (e.g. TS cond 3, none)."),readln(ocond),
    assertz(obs(Olat,Olong,Otime,Oday,Oyear,Oint,Ospeed)),
    asserta(posit_list([[Otime,Oday,Olat,Olong,Oint]|[]])),
    asserta(cond_list([Ocond])),
    asserta(gust_list([Ogust])).

 input_forecast:-
    write("Forecast storm latitude  - "), readreal (Flat),
    write("Forecast storm longitude - "), readreal(Flong),
    write("Forecast time - "), readreal(Ftime),
    write("Forecast Julian day - "), readreal(Fday),
    write("Forecast year - "),readint(Fyear),
    write("Forecast max surface winds - "), readreal(Fint),
```

```
   assertz(forecast(Flat,Flong,Ftime,Fday,Fyear,Fint)), nl,
   obs_time(Vtime,Vday),
   time_diff(Ftime,Fday,Vtime,Vday,Time_diff),
   assertz(forecast_int(Time_diff)),
   retract(forecast_int(_)),
   clearwindow.

menu(0):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    1) Determine condition based on your forecast,"),nl,
   write("    2) Let program modify forecast and pick condition,"),nl,
   write("    6) Extrapolate observation along closest statistically "),nl,
   write("       preferred path across the Philippines,"),nl,
   write("    7) Extrapolate observation along a user chosen preferred"),nl,
   write("       path,"),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
menu(1):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    1) Determine condition based on your forecast,"),nl,
   write("    2) Let program modify your forecast and pick condition,"),nl,
   write("    3) Replace observation with last forecast and observed"),nl,
   write("       speed, and input a later forecast,"),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
menu(2):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    2) Let program modify forecast and pick condition,"),nl,
   write("    4) Replace observation with modified forecast and"),nl,
   write("       input a later forecast to be modified,"),nl,
   write("    5) Generate a persistance plus modification track "),nl,
   write("       with no further forecast input,"),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
menu(3):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    3) Replace observation with your forecast and observed"),nl,
   write("       speed, and input a later forecast,"),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
```

```
menu(4):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    4) Replace observation with modified forecast and"),nl,
   write("       input a later forecast to be modified, "),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
menu(5):-
   clearwindow,
   write("Will you want to:"),nl,
   write("    5) Generate a persistance plus modification track "),nl,
   write("       with no further forecast input,"),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).
menu(X):-
   X>=6,
   clearwindow,
   write("Will you want to:"),nl,
   write("    8) Continue the extrapolation along a preferred path, "),nl,
   write(" or 9) Exit program?"),nl,
   readint(Ans),
   clearwindow,
   do(Ans),
   menu(Ans).

do(1):-
   input_forecast,
   retract(mod_forecast(_,_,_,_,_,_,_)),
   retract(forecast(_,_,_,_,_,_)),
   forecast(A,B,C,D,E,F),
   obs(_,_,_,_,_,_,S),
   assertz(mod_forecast(A,B,C,D,E,F,S)),
   condition(Condition),
   store([C,D,A,B,F],Condition),
   write("Based on your forecast for ",C," hours on day ",D,","),nl,
   write("recommend setting ", Condition),nl,
   write("(Press any key to continue.)"), readchar(_).
do(2):-
   input_forecast,
   retract(forecast(_,_,_,_,_,_)),
   retract(mod_forecast(_,_,_,_,_,_,_)),
   modify_forecast,
   mod_forecast(Lat,Long,TT,DD,_,Intensity,Speed),
   forecast(A,B,_,_,_,C), obs(_,_,_,_,_,_,D),
   writef("Modified latitude is  %6.2N vice %6.2N.",Lat,A),nl,
   writef("Modified longitude is %6.2E vice %6.2E.",Long,B),nl,
   writef("Modified intensity is %6.2kts vice %6.2kts.",Intensity,C),nl,
   writef("Modified speed is     %5.2kts vice %5.2kts.",Speed,D),nl,
   condition(Condition),
   store([TT,DD,Lat,Long,Intensity],Condition),
```

```
   write("at ",TT," hours on day ",DD,"."),nl,
   write("Based on this modified forecast position,"),nl,
   write("recommend setting ", Condition),nl,
   write("(Press any key to continue.)"), readchar(_).
do(3):-
   retract(mod_forecast(_,_,_,_,_,_,_)),
   forecast(A,B,C,D,E,F), obs(_,_,_,_,_,_,S),
   assertz(obs(A,B,C,D,E,F,S)), retract(obs(_,_,_,_,_,_,_)),
   assertz(obs_time(C,D)), retract(obs_time(_,_)),
   write("Your last forecast was - "),nl,
   writef("   Latitude - %6.2", A), nl,
   writef("  Longitude - %6.2", B), nl,
   writef("  Intensity - %6.2", F), nl,
   writef("      Speed - %6.2", S), nl,
   writef("   Time/Day _ %4/%3",C,D),nl,
   retract(forecast(_,_,_,_,_,_)), input_forecast,
   forecast(Q,R,T,U,V,W),
   obs(_,_,_,_,_,_,S),
   assertz(mod_forecast(Q,R,T,U,V,W,S)),
   condition(Condition),
   store([T,U,Q,R,W],Condition),
   write("Based on your forecast for ",T," hours on day ",U),nl,
   write("recommend setting ", Condition),nl,
   write("(Press any key to continue.)"), readchar(_).
do(4):-
   retract(forecast(_,_,_,_,_,_)),
   mod_forecast(A,B,C,D,E,F,G),
   assertz(obs(A,B,C,D,E,F,G)), retract(obs(_,_,_,_,_,_,_)),
   assertz(obs_time(C,D)), retract(obs_time(_,_)),
   retract(mod_forecast(_,_,_,_,_,_,_)),
   input_forecast, modify_forecast,
   mod_forecast(Lat,Long,_,_,_,Intensity,Speed),
   forecast(W,X,TT,DD,_,Y),
   writef("Modified latitude is  %6.2N vice %6.2N",Lat,W),nl,
   writef("Modified longitude is %6.2E vice %6.2E",Long,X),nl,
   writef("Modified intensity is %5.1kts vice %5.1kts",Intensity,Y),nl,
   writef("Modified speed is     %4.1kts vice %4.1kts",Speed,G),nl,
   condition(Condition),
   store([TT,DD,Lat,Long,Intensity],Condition),
   write("at ",TT," hours on day ",DD,"."),nl,
   write("Based on this modified forecast position,"),nl,
   write("recommend setting ", Condition),nl,
   write("(Press any key to continue.)"), readchar(_).
do(5):-
   retract(forecast(_,_,_,_,_,_)),
   obs(Olat,Olong,_,_,_,_,_),
   mod_forecast(Mlat,Mlong,Mtime,Mday,Myear,Mint,Mspeed),
   range(Olat,Olong,Mlat,Mlong,Dist),
   heading(Olat,Olong,Mlat,Mlong,Dist,Angle),
   writef("Heading is %3.0 degrees.", Angle),nl,
   writef("Distance is %5.1 n.m..", Dist),nl,
   xtrap(Mlat,Mlong,Angle,Dist,Flat,Flong),
   forecast_int(Diff),
   D = Diff*100,
   newtime(Mtime,Mday,Myear,Ntime,Nday,Nyear,D),
   assertz(obs(Mlat,Mlong,Mtime,Mday,Myear,Mint,Mspeed)),
```

```
    retract(obs(_,_,_,_,_,_,_,_)),
    retract(mod_forecast(_,_,_,_,_,_,_,_)),
    assertz(forecast(Flat,Flong,Ntime,Nday,Nyear,Mint)),
    modify_forecast,
    mod_forecast(Lat,Long,_,_,_,Intensity,Speed),
    writef("Modified latitude is  %6.2N",Lat),nl,
    writef("Modified longitude is %6.2E",Long),nl,
    writef("Modified intensity is %6.2kts",Intensity),nl,
    writef("Modified speed is     %5.2kts",Speed),nl,
    condition(Condition),
    store([Ntime,Nday,Lat,Long,Intensity],Condition),
    write("at ",Ntime," hours on day ",Nday,"."),nl,
    write("Based on this modified persistance position,"),nl,
    write("recommend ", Condition),nl,
    write("(Press any key to continue.)"), readchar(_).
do(6):-
    obs(Olat,Olong,_,_,_,_,_,_),
    findall(A,path_point(A,Olong),L),
    closest_path(L,Path,D,Olat,Olong),
    assertz(p(Path)), retract(p(_)),
    writef("The distance to path %1.0 is %5.1.",Path,D),nl,nl,
    write("Input extrapolation time step in hours (e.g. 6)."),
    readreal(Diff),nl,
    assertz(forecast_int(Diff)), retract(forecast_int(_)),
    do_more(Path).
do(7):-
    obs(Olat,Olong,_,_,_,_,_,_),
    write("Input desired path, from 1(North) to 5(South). "),
    readreal(Path),
    assertz(p(Path)), retract(p(_)),
    path(Path,_,Plat,Olong),
    range(Olat,Olong,Plat,Olong,D),
    writef("The distance to path %1.0 is %5.1.",Path,D),nl,
    write("Input extrapolation time step in hours (e.g. 6)."),
    readreal(Diff),nl,
    assertz(forecast_int(Diff)), retract(forecast_int(_)),
    do_more(Path).
do(8):-
    p(Path),
    do_more(Path).
do(9):-
    posit_list(Plist),cond_list(Clist), gust_list(Glist),
    write("Time Day  Lat  Long   Int  Gust   Resulting Condition"),nl,
    write_output(Plist,Glist,Clist),
    write("Do you wish to store this output to disk?"),readln(Res),
    store_output(Res),
    write("Press any key to exit program."),readchar(_),
    retract(posit_list(_)),
    retract(cond_list(_)),
    retract(gust_list(_)),
    retract(obs(_,_,_,_,_,_,_,_)),
    retract(obs_time(_,_)),
    retract(base_time(_,_,_)),
    retract(forecast(_,_,_,_,_,_)),
    retract(mod_forecast(_,_,_,_,_,_,_,_)),
    retract(forecast_int(_)),
```

```
   retract(p(_)),
   exit.

do_more(Path):-
   obs(_,Olong,Otime,Oday,Oyear,Oint,Ospeed),
   path(Path,Theta,Plat,Olong),
   forecast_int(Diff),
   Dist = Diff * Ospeed,
   xtrap(Plat,Olong,Theta,Dist,Flat,Flong),
   D = Diff * 100,
   newtime(Otime,Oday,Oyear,Ftime,Fday,Fyear,D),
   assertz(forecast(Flat,Flong,Ftime,Fday,Fyear,Oint)),
   retract(forecast(_,_,_,_,_,_)),
   modify_forecast,
   retract(mod_forecast(_,_,_,_,_,_,_)),
   mod_forecast(_,Mlong,_,_,_,Mint,Mspeed),
   path(Path,_,Lat,Mlong),
   assertz(mod_forecast(Lat,Mlong,Ftime,Fday,Fyear,Mint,Mspeed)),
   retract(mod_forecast(_,_,_,_,_,_,_,_)),
   writef("Modified latitude is  %6.2N",Lat),nl,
   writef("Modified longitude is %6.2E",Mlong),nl,
   writef("Modified intensity is %6.2kts",Mint),nl,
   writef("Modified speed is     %5.2kts",Mspeed),nl,
   condition(Condition),
   store([Ftime,Fday,Lat,Mlong,Mint],Condition),
   write("at ",Ftime, " hours on day ",Fday,"."),nl,
   write("Based on this extrapolated position,"),nl,
   write("recommend ", Condition),nl,
   write("Press any key to continue."), readchar(_),
   assertz(obs(Lat,Mlong,Ftime,Fday,Fyear,Mint,Mspeed)),
   retract(obs(_,_,_,_,_,_,_)),
   clearwindow.

store_output("n").
store_output("y"):-
   write("Input storm identification."), readln(Storm),
   write("Input run type (e.g., Unmod, Mod, or Climo)."), readln(Test),
   openappend(resultfile,"test.run"),
   writedevice(resultfile),
   write(Storm,Test),nl,
   posit_list(Plist),cond_list(Clist), gust_list(Glist),
   write("Time Day  Lat  Long   Int  Gust   Resulting Condition"),nl,
   write_output(Plist,Glist,Clist),nl,
   closefile(resultfile),
   writedevice(screen).

write_output([],[],[]).
write_output([A|B],[C|D],[E|F]):-
   write_list(A,C),write(E),nl,write_output(B,D,F).
write_list([U,V,W,X,Y],Z):-
   writef("%4 %3 %4.1 %5.1 %5.1 %5.1    ",U,V,W,X,Y,Z).

append([],L,[L]).
append([X|L1],L2,[X|L3]):- append(L1,L2,L3).

store([A,B,C,D,E],F):-
```

```
    posit_list(Alist), append(Alist,[A,B,C,D,E],Blist),
    assertz(posit_list(Blist)),retract(posit_list(_)),
    cond_list(Clist), append(Clist,F,Dlist),
    assertz(cond_list(Dlist)),retract(cond_list(_)).

/* Path defines the statistically preferred paths defined in Sikora.      */
/* Second argument is the heading along the path.                         */
 path(1,291.8,Lat,Long):- Lat = -0.40*Long + 66.8.
 path(2,286.7,Lat,Long):- Lat = -0.30*Long + 52.7.
 path(3,272.9,Lat,Long):- Long>=123.6, Lat = -0.05*Long + 18.8.
 path(3,293.8,Lat,Long):- Long<123.6,  Lat = -0.44*Long + 67.1.
 path(4,275.1,Lat,Long):- Long>=125.5, Lat = -0.09*Long + 23.0.
 path(4,286.2,Lat,Long):- Long<125.5,  Lat = -0.29*Long + 47.7.
 path(5,272.9,Lat,Long):- Long>=125.7, Lat = -0.05*Long + 15.4.
 path(5,278.0,Lat,Long):- Long<125.7,  Lat = -0.14*Long + 26.9.

 /* Closest_path finds the preferred path closest to the given observation */
 /* by sorting the list of path positions corresponding to the obs long    */
 /* as created using the path_point rule.                                  */
 path_point(A,Long):- path(B,_,C,Long), A=[B,C,Long].
 closest_path([[Path,Lat,Long]|[]],Path,Range,Olat,Olong):-
    range(Olat,Olong,Lat,Long,Range).
 closest_path([[Path,Lat,Long]|L],Path,Range,Olat,Olong):-
    range(Olat,Olong,Lat,Long,Range),
    closest_path(L,_,Lrange,Olat,Olong),
    Range<Lrange.
 closest_path([[_,Lat,Long]|L],Lpath,Lrange,Olat,Olong):-
    range(Olat,Olong,Lat,Long,Range),
    closest_path(L,Lpath,Lrange,Olat,Olong),
    not(Range<Lrange).

/* Heading calculates the bearing from one position to another given */
/* the distance, based on great circle.                              */
 heading(Lata,Longa,Latb,Longa,_,0.0):- Lata<Latb,!.
 heading(Lata,Longa,Latb,Longa,_,180.0):- Lata>Latb,!.
 heading(Lata,Longa,Lata,Longb,_,90.0):- Longa<Longb,!.
 heading(Lata,Longa,Lata,Longb,_,270.0):- Longa>Longb,!.
 heading(Lata,Longa,Latb,Longb,Dist,Angle):- C = 0.0174533,
    sin(C*(Longa-Longb))<0,
    D = Dist/60,
    X=(sin(C*Latb) - sin(C*Lata)*cos(C*D))/(sin(C*D)*cos(C*Lata)),
    V = 1.0 - X*X, Rv = sqrt(V),
    Angle = (1.0/C)*(1.5708 - arctan(X/Rv)).
 heading(Lata,Longa,Latb,Longb,Dist,Angle):- C = 0.0174533,
    sin(C*(Longa-Longb))>=0,
    D = Dist/60,
    X=(sin(C*Latb) - sin(C*Lata)*cos(C*D))/(sin(C*D)*cos(C*Lata)),
    V = 1.0 - X*X, Rv = sqrt(V),
    Angle = 360.0 - (1.0/C)*(1.5708 - arctan(X/Rv)).

 xtrap(Lat,Long,Course,Dist,Latb,Longb):-  C = 0.0174533,
    Latb = Lat + (Dist*cos(Course*C))/60.0,
    Longb = Long + (Dist*sin(Course*C)*cos(Lat*C))/60.0.

/* Newtime adds a time interval in 100's of hours to a given time    */
 newtime(Otime,Oday,Oyear,Ntime,Oday,Oyear,Diff):-
```

```
    Ntime=Otime+Diff, Ntime <2400,!.
 newtime(Otime,Oday,Oyear,Ntime,Nday,Oyear,Diff):-
    Ntime=Otime+Diff-2400, Nday=Oday+1, Nday<=365,!.
 newtime(Otime,Oday,Oyear,Ntime,Nday,Oyear,Diff):-
    Ntime=Otime+Diff-2400, Nday=Oday+1, Nday=366,
    Oyear mod 4 = 0, Oyear mod 100<>0,!.
 newtime(Otime,Oday,Oyear,Ntime,Nday,Nyear,Diff):-
    Oyear mod 4 = 0, Oyear mod 100<>0, Day = Oday - 366,
    Nyear = Oyear+1, newtime(Otime,Day,Nyear,Ntime,Nday,Nyear,Diff).
 newtime(Otime,Oday,Oyear,Ntime,Nday,Nyear,Diff):-
    Day = Oday-365, Nyear=Oyear+1,
    newtime(Otime,Day,Nyear,Ntime,Nday,Nyear,Diff).

/*  Time, day and year of the observation are used to     */
/*  calculate the time difference between observation     */
/*  and forecast.  Program is based on winds at forecast  */
/*  time at Cubi Pt.                                      */
/*  condition and result predicates form the steps that  */
/*  derive the expected condition code at Cubi based on  */
/*  expected winds and time until their arrival.          */
 condition(Condition):-
    wind(Gust),
    base_time(Time,Day,_),
    mod_forecast(_,_,Ftime,Fday,_,_,_),
    time_diff(Ftime,Fday,Time,Day,Time_diff),
    result(Condition,Gust,Time_diff),
    gust_list(Alist),
    append(Alist,Gust,Blist),
    assertz(gust_list(Blist)),
    retract(gust_list(_)).

 result("tropical storm condition 1",Gust,Time_diff):-
    Gust>=35,Time_diff<=12,!.
 result("tropical storm condition 2",Gust,Time_diff):-
    Gust>=35,Time_diff<=24,!.
 result("tropical storm condition 3",Gust,Time_diff):-
    Gust>=35,Time_diff<=48,!.
 result("condition 4 (winds)",Gust,_):-Gust<35.
 result("condition 4 (time)",_,Time_diff):-Time_diff>=48.

/*   wind predicates calculate the expected maximum gust  */
/*   speeds based on the max storm winds and the study    */
/*   data from Jarrell and Englebretson.                   */
 wind(0.0):-mod_forecast(Lat,Long,Time,Day,_,_,_),
    radius_to_cubi(Lat,Long,Radius),Radius>360,
    write("Storm is outside study area at ",Time," on  day ",Day),nl.
 wind(Gust):-mod_forecast(_,_,_,_,_,Intensity,_),
    Intensity<64,ts_factor(Factor),Gust=Intensity*Factor.
 wind(Gust):-mod_forecast(_,_,_,_,_,Intensity,_),
    Intensity>=64,tc_factor(Factor),Gust=Intensity*Factor.

/*  ts_factor and tc_factor work to find the data fact    */
/*  for the closest segment in the study area to the      */
/*  forecast storm position, and get the max gust         */
/*  multiplying factor from that fact.  The findall       */
/*  predicates form a list of all segment centers within  */
```

```
/*  a distance defined by the seg_dist rule to the storm. */
/*  Then the closest_seg rules sort through this list to  */
/*  find the one segment center closest to the storm.     */
/*  This segment number is printed out for information.   */
 ts_factor(Factor):-findall(A,seg_dist_ts(A,_),L),
    closest_seg(L,Clat,Clong,_),ts(_,[Clat,Clong],Sus_max,_)
    Factor=1.5*Sus_max.
 tc_factor(Factor):-findall(A,seg_dist_tc(A,_),L),
    closest_seg(L,Clat,Clong,_),tc(_,[Clat,Clong],Sus_max,_),
    Factor=1.5*Sus_max.

 closest_seg([[Clat,Clong]|[]],Clat,Clong,Dist):-dist(Clat,Clong,Dist).
 closest_seg([[Clat,Clong]|L],Clat,Clong,Dist):-dist(Clat,Clong,Dist),
    closest_seg(L,_,_,Adist),Dist<Adist.
 closest_seg([[Clat,Clong]|L],Alat,Along,Adist):-dist(Clat,Clong,Dist),
    closest_seg(L,Alat,Along,Adist),not(Dist<Adist).

 seg_dist_ts([Lat,Long],Dist):-ts(_,[Lat,Long],_,_),
    dist(Lat,Long,Dist),Dist<80.
 seg_dist_tc([Lat,Long],Dist):-tc(_,[Lat,Long],_,_),
    dist(Lat,Long,Dist),Dist<80.

 /*  dist calculates the distance from the forecast       */
 /*  position to any other lat/long, using the range rule.*/
 dist(Clat,Clong,D):-mod_forecast(Lat,Long,_,_,_,_,_),range(Clat,Clong,Lat,Long,
D).
 /*  radius_to_cubi finds the distance from a lat/long to Cubi Pt. */
 radius_to_cubi(Lat,Long,D):-range(Lat,Long,14.8,120.3,D).
 /*  range does the mathematics of great circle distance finding.  */
 range(Clat,Clong,Clat,Clong,0).
 range(Clat,Clong,Lat,Long,D):-C=0.0174532,
    X=sin(Lat*C)*sin(Clat*C)+cos(Lat*C)*cos(Clat*C)*cos((Clong-Long)*C),
    V=1.0-(X*X),Rv=sqrt(V),D=(60/C)*(1.5708-arctan(X/Rv)).

/*  Time, day and year of the observation are used to     */
/*  calculate the time difference between observation     */
/*  and forecast.  Program is based on winds at forecast  */
/*  time at Cubi Pt.                                      */
/*  time_diff calculates the time difference in hours between  */
/*  any two Julian days and times, with corrections for leap year.*/
 time_diff(Time2,Day2,Time1,Day1,Diff):-Day2-Day1>=0,
    Diff=24*(Day2-Day1)+(Time2-Time1)/100.
 time_diff(Time2,Day2,Time1,Day1,Diff):-Day2-Day1<0,
    base_time(_,_,Vyear),
    Vyear mod 4=0,Vyear mod 100<>0,
    New_day2=Day2+366,time_diff(Time2,New_day2,Time1,Day1,Diff),!.
 time_diff(Time2,Day2,Time1,Day1,Diff):-Day2-Day1<0,New_day2=Day2+365,
    time_diff(Time2,New_day2,Time1,Day1,Diff).

/* CPA calculates time to CPA, assuming westward storm travel. */
 cpa(Olat,Olong,Ospeed):- Olat>=14.5, Olong>=120.3,
    range(Olat,Olong,Olat,120.3,D), Seafall = D/Ospeed,
    assertz(seafall(Seafall)),
    writef("Estimate %5.2 hours until storm enters South China Sea.",
    Seafall),nl.
 cpa(Olat,Olong,_):-Olat>=14.5, Olong<120.3,
```

```
   write("How many hours since storm crossed into the South China Sea?"),nl,
   write("(negative if unknown)"), readreal(Q), Q>=0.0, NQ=-Q,
   assertz(seafall(NQ)), nl, !.
cpa(Olat,Olong,Ospeed):- Olat>=14.5, Olong<120.3,
   range(Olat,Olong,Olat,120.3,D), Seafall=-D/Ospeed,
   assertz(seafall(Seafall)), X=-Seafall,
   writef("Estimate %5.2 hours since storm entered the South China Sea."
   , X), nl.
cpa(Olat,Olong,Ospeed):- Olat<14.5, Olong>=122.0,
   range(Olat,Olong,Olat,122.0,D), Seafall=D/Ospeed,
   assertz(seafall(Seafall)),
   writef("Estimate %5.2 hours until storm enters Sulu Sea.",Seafall), nl.
cpa(Olat,Olong,_):- Olat<14.5, Olong<122.0,
   write("How many hours since storm entered the Sulu Sea?"),
   write("(negative if unknown)"), readreal(Q),
   Q>=0.0, NQ=-Q, assertz(seafall(NQ)),nl, !.
cpa(Olat,Olong,Ospeed):- Olat<14.5, Olong<122.0,
   range(Olat,Olong,Olat,122.0,D), Seafall=-D/Ospeed,
   assertz(seafall(Seafall)), X=-Seafall,
   writef("Estimate %5.2 hours since storm entered the Sulu Sea.",
   X), nl.

/* modify_forecast is the instruction that controls the creation  */
/* of the mod_forecast fact.  A new intensity is calculated based */
/* on the observed intensity and the time to CPA at Cubi, as      */
/* calculated in Brand and Blelloch study.                        */
/* A modified forecast position is calculated based on the change */
/* in storm speed as estimated using Sikora, and                  */
/* converting this to a moodified position, assuming the same     */
/* heading.  Then, a north/south correction is added due to the   */
/* forecast error tendency noted in B and B.                      */
modify_forecast:-
   obs(Olat,Olong,_,_,_,Oint,Ospeed),
   forecast(Flat,Flong,Ftime,Fday,Fyear,_),
   forecast_int(Diff),
   cpa(Olat,Olong,Ospeed),
   seafall(Seafall),
   ave_strength(Seafall, Olat, Oint),
   strength(Intensity,Oint,Diff,Seafall,Olat),
   posit(Lat,Long,Speed,Oint,Diff,Seafall,Flat,Flong,Olat,Olong,Ospeed),
   retract(response(_)),
   Int = Seafall - Diff, /* interval from forecast posit to 120.3E */
   xtrack(Mlat,Lat,Int,Diff),
   assertz(mod_forecast(Mlat,Long,Ftime,Fday,Fyear,Intensity,Speed)),
   retract(seafall(_)).

ave_strength(Seafall,Olat,Oint):- Olat>14.5, Oint>80,Seafall<8.2,
   write("Was the ave. storm intensity for 24 hours before ",
   "hitting land >= 80 kts?"), nl, write(" (y, n, or u) "),
   readln(Ans), assertz(response(Ans)).
ave_strength(Seafall,Olat,Oint):-Olat>14.5, Oint<80, Seafall<9.1,
   write("Was the ave. storm intensity for 24 hours before ",
   "hitting land >= 80 kts?"), nl, write(" (y, n, or u) "),
   readln(Ans), assertz(response(Ans)).
ave_strength(Seafall,Olat,Oint):-Olat<14.5, Oint>80, Seafall<26.4,
   write("Was the ave. storm intensity for 24 hours before ",
```

```
    "hitting land >= 80 kts?"), nl, write(" (y, n, or u) "),
    readln(Ans), assertz(response(Ans)).
 ave_strength(Seafall,Olat,Oint):-Olat<14.5, Oint<80, Seafall<18.4,
    write("Was the ave. storm intensity for 24 hours before ",
    "hitting land >= 80 kts?"), nl, write(" (y, n, or u) "),
    readln(Ans), assertz(response(Ans)).
 ave_strength(_):-  assertz(response("u")).

/* posit uses the modified speed from B and B to generate a new   */
/* lat/long.  Correction for land-induced forecast error N/S is   */
/* accounted for in xtrack.                                       */
 posit(Lat,Long,Speed,Oint,Diff,Seafall,Flat,Flong,Olat,Olong,Ospeed):-
    speed(Factor,Oint,Diff,Seafall,Olat,Ospeed),
    Lat = Olat + Factor*(Flat - Olat),
    Long = Olong + Factor*(Flong - Olong),
    Speed = Ospeed * Factor.

/* speed calculates the ratio of terrain corrected SOA to observed */
/* storm speed for use in the posit rule.  Data is from the curves */
/* presented in SIKORA!!                                           */
/* First 10 rules are for a northern transit, above 14.5N          */
 speed(Factor,Oint,Diff,Seafall,Olat,_):-Olat>=14.5, Oint<80, Seafall>=40,
    Factor = 1.0 - 0.0084*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat>=14.5, Oint<80, Seafall>=18,
    Factor = 1.0 - 0.00313*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat>=14.5, Oint<80, Seafall>=8.2,
    Factor = 1.0 + 0.2173*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat>=14.5, Oint<80, Seafall>=-16,
    response(Ans), not(Ans="y"),
    Factor = 1.0 - 0.00738*Diff.
 speed(Factor,Oint,_,Seafall,Olat,_):- Olat>=14.5, Oint<80, Seafall<-16,
    response(Ans),  not(Ans="y"),
    Factor = 1.0, !.
 speed(Factor,_,_,Seafall,Olat,_):- Olat>=14.5, Seafall>=45,
    Factor = 1.0, !.
 speed(Factor,_,Diff,Seafall,Olat,_):- Olat>=14.5, Seafall>=27,
    Factor = 1.0 - 0.00608*Diff, !.
 speed(Factor,_,Diff,Seafall,Olat,_):- Olat>=14.5, Seafall>=8.2,
    Factor = 1.0 + 0.00902*Diff, !.
 speed(Factor,_,_,Seafall,Olat,_):- Olat>=14.5, Seafall>=0,
    Factor = 1.0.
 speed(Factor,_,Diff,Seafall,Olat,_):- Olat>=14.5, Seafall<0,
    Factor = 1.0 - 0.00996*Diff.
/*these 14 rules are for a southern transit, below 14.5N          */
  speed(Factor,Oint,Diff,Seafall,Olat,Ospeed):- Olat<14.5, Oint<80,
    Ospeed<11, Seafall>=49,
    Factor = 1.0 - 0.01622*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,Ospeed):- Olat<14.5, Oint<80,
    Ospeed<11, Seafall>=20.9,
    Factor = 1.0 + 0.02121*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,Ospeed):- Olat<14.5, Oint<80,
    Ospeed<11, Seafall>=8.2,
    Factor = 1.0 - 0.00706*Diff, !.
 speed(Factor,Oint,Diff,Seafall,Olat,Ospeed):- Olat<14.5, Oint<80,
    Ospeed<11, Seafall>=-8, response(Ans), not(Ans="y"),
    Factor = 1.0 - 0.00706*Diff, !.
```

```
speed(Factor,Oint,Diff,Seafall,Olat,Ospeed):- Olat<14.5, Oint<80,
   Ospeed<11, Seafall<-8, response(Ans), not(Ans="y"),
   Factor = 1.0 + 0.00962*Diff.
speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall>=45,
   Factor = 1.0 + 0.00972*Diff, !.
speed(Factor,Oint,_,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall>=27,
   Factor = 1.0.
speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall>=18,
   Factor = 1.0 + 0.01126*Diff, !.
speed(Factor,Oint,_,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall>=8.2,
   Factor = 1.0, !.
speed(Factor,Oint,_,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall>=-3,  response(Ans), not(Ans="y"),
   Factor = 1.0.
speed(Factor,Oint,Diff,Seafall,Olat,_):- Olat<14.5, Oint<80,
   Seafall<-3, response(Ans), not(Ans="y"),
   Factor = 1.0 - 0.00730*Diff.
speed(Factor,_,Diff,Seafall,Olat,_):- Olat<14.5, Seafall>=39,
   Factor = 1.0 - 0.00970*Diff, !.
speed(Factor,_,Diff,Seafall,Olat,_):- Olat<14.5, Seafall>=0,
   Factor = 1.0 + 0.00675*Diff.
speed(Factor,_,Diff,Seafall,Olat,_):- Olat<14.5, Seafall<0,
   Factor = 1.0 - 0.01042*Diff.

/* strength calculates terrain modified intensity based on the      */
/* observed intensity iaw SIKORA!!                                  */
strength(Int,Oint,Diff,Seafall,Olat):- Olat>=14.5,Oint<80, Seafall>=8.2,
   Int = Oint*(1.0 + 0.0170*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat>=14.5,Oint<80, Seafall>=0.0,
   response(Ans), not(Ans="y"),
   Int = Oint*(1.0 - 0.02121*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat>=14.5,Oint<80, Seafall<0.0,
   response(Ans), not(Ans="y"),
   Int = Oint*(1.0 + 0.00731*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat>=14.5, Seafall>=26,
   Int = Oint*(1.0 + 0.01151*Diff), !.
strength(Int,Oint,_,Seafall,Olat):- Olat>=14.5, Seafall>=9,
   Int = Oint, !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat>=14.5, Seafall>=0,
   Int = Oint*(1.0 - 0.03933*Diff).
strength(Int,Oint,_,Seafall,Olat):- Olat>=14.5, Seafall<0,
   Int = Oint.
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Oint<80, Seafall>=20.9,
   Int = Oint*(1.0 + 0.01964*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Oint<80, Seafall>=8.2,
   Int = Oint*(1.0 - 0.00844*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Oint<80, Seafall>=0,
   response(Ans), not(Ans="y"),
   Int = Oint*(1.0 - 0.00844*Diff), !.
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Oint<80, Seafall<0,
   response(Ans), not(Ans="y"),
   Int = Oint*(1.0 + 0.00818*Diff), !.
```

```
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Seafall>=38,
   Int = Oint*(1.0 + 0.01655*Diff), !.
strength(Int,Oint,_,Seafall,Olat):- Olat<14.5, Seafall>=20,
   Int = Oint.
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Seafall>=-3,
   Int = Oint*(1.0 - 0.0219*Diff).
strength(Int,Oint,Diff,Seafall,Olat):- Olat<14.5, Seafall<-3,
   Int = Oint*(1.0 + 0.0035*Diff).

/* xtrack adds in the N/S correction over and west of the islands  */
/* as noted in B and B.  Correction is added onto the position     */
/* after it is modified for the new speed along the forecast line  */
/* of advance (in 'position').                                     */
/* Formula used to get average correction per forecast based on B&B*/
/* is:                                                             */
/*   [(correction/transit)*(hours/forecast)] / [hours/transit]     */
xtrack(Mlat,Lat,Int,Diff):-
   Int>0, Int<=13, Diff<=13,
   Mlat = Lat + (18.3/60.0)*Diff/13.
xtrack(Mlat,Lat,Int,Diff):-
   Int>0, Int<=13, Diff>13,
   Mlat = Lat + 18.3/60.
xtrack(Mlat,Lat,Int,Diff):-  /* This clause covers when the forecast  */
   Int<=0, Int>24, Diff<=24, /* interval skips over the 0-13 hour     */
   Int + Diff>13,            /* region (land) entirely.               */
   Mlat = Lat - (14.4/60.0)*Diff/24 + 18.3/60,!.
xtrack(Mlat,Lat,Int,Diff):-
   Int<=0, Int>-24, Diff<=24,
   Mlat = Lat - (14.1/60.0)*Diff/24.
xtrack(Mlat,Lat,Int,Diff):-
   Int<=0, Int>24,Diff>24,
   Mlat = Lat - 14.1/60.
xtrack(Lat,Lat,_,_).
```

REFERENCES

Bender, M. A., R. E. Tuleya and Y. Kurihara, 1987: A numerical study of the effect of island terrain on tropical cyclones. *Mon. Wea. Rev., 115*, 130-155.

Brand, S., and J. W. Blelloch, 1973: Change in the characteristics of typhoons crossing the Philippines. *J. Appl. Meteor., 12*, 104-109.

_____ and _____, 1974: Change in the characteristics of typhoons crossing the island of Taiwan. *Mon. Wea. Rev., 102*, 708-713.

Chang, S. W., 1982: The orographic effects induced by an island mountain range on propagating tropical cyclones. *Mon. Wea. Rev., 110*, 1255-1270.

Clocksin, W., and C. Mellish, 1984: *Programming in PROLOG*, Springer-Verlag, 2 ed., 297 pp.

Douglass, J. A., 1975: An evaluation of the harbors of Subic Bay and Manila, Republic of the Philippines, as typhoon havens. ENVPREDRSCHFAC Technical Paper No. 13-75, 115 pp.

Elsberry, R. L., W. M. Frank, G. J. Holland, J. D. Jarrell and R. L. Southern, 1987: *A Global View of Tropical Cyclones*. Office of Naval Research publication (available from R. L. Elsberry, Naval Postgraduate School, Monterey, CA, 93943), 185 pp.

Forecaster's Handbook - NAS Cubi Point, Philippines, 1984: Naval Oceanography Command Facility, Cubi Point, Philippines, 116 pp.

Jarrell, J. D. and R. E. Englebretson, 1982: Forecast aids for predicting tropical cyclone associated gusts and sustained winds for Cubi Point, Philippines. NAVENVPREDRSCHFAC Contractor Report 82-10, 14 pp.

Peak, J. E., 1987: A prototype expert system for shipboard obscuration prediction, Internal Report, Martin Marietta Data Systems (available from Naval Environmental Prediction Research Facility, Monterey, CA, 93943), 67 pp.

Sandgathe, S. A., 1987: Opportunities for tropical cyclone motion research in the Northwest Pacific region. Tech. Rep. NPS-63-87-006, Naval Postgraduate School, Monterey, Ca., 93943, 35 pp.

Sikora, C. R., 1976: A reevaluation of the changes in speed and intensity of tropical cyclones crossing the Philippines. FLEWEACEN Tech Note: JTWC 76-2, 11 pp.

U. S. Fleet Weather Central/Joint Typhoon Warning Center: *Annual Tropical Cyclone Reports*, 1984-1987.

Wang, S. T., 1980: Prediction of the behavior and strength of typhoons in Taiwan and its vicinity. Res. Rep. 18, NSC-NSC-67M-0202-05(01), 1-100.

Wu, T., and S. Wang, 1983: Circulation and track changes of typhoons encountering the central mountain range of Taiwan: Field observations and laboratory experiments. *Proc. of CCNAA-AIT Joint Seminar on Monsoon and Tropical Meteorology*, 117-128.

Zubrick, S., 1985: Validation of a weather forecasting expert system. Presented at the Machine Intelligence Workshop 11, Loch Lomond, Scotland, March 25-29, 1985, 21 pp.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000	1
4.	Oceanographer of the Navy Naval Observatory 34th and Massachusetts Avenue NW Washington, DC 20390-5000	1
5.	Commander Naval Oceanography Command NSTL, MS 39522-5000	1
6.	Commanding Officer Fleet Numerical Oceanography Center Monterey, CA 93943-5005	1
7.	Chairman, Code 63Rd Department of Meteorology Naval Postgraduate School Monterey, CA 93943-5000	1
8.	Chairman, Code 68Co Department of Oceanography Naval Postgraduate School Monterey, CA 93943-5000	1
9.	Professor Russell L. Elsberry, Code 63Es Department of Meteorology Naval Postgraduate School Monterey, CA 93943-5000	5
10.	Dr. G. J. Holland BMRC P.O. Box 1289K Melbourne Vic 3001 Australia	1

11. LT Bruce M. Hagaman — 2
USS Saratoga CV-60
FPO New York, NY 09543-2740

12. Commanding Officer — 1
Naval Environmental Prediction
Research Facility
Monterey, CA 93943-5006

13. Chairman, Oceanography Department — 1
U.S. Naval Academy
Annapolis, MD 21402-5000

14. Mr. Ted Tsui — 1
Naval Environmental Prediction
Research Facility
Monterey, CA 93943-5006

15. Director — 1
Joint Typhoon Warning Center
COMNAVMARIANAS Box 17
FPO San Francisco, CA 96630

16. Library Acquisitions — 1
National Center for Atmospheric Research
P.O. Box 3000
Boulder, CO 80307-5000

17. Commanding Officer — 1
Naval Oceanography Command Facility
Box 63 NAS Cubi Point
FPO San Francisco, CA 96654-2909

18. Mr. Jim Peak — 1
Naval Environmental Prediction
Research Facility
Monterey, CA 93943-5006

19. Mr. Jerry Jarrell — 1
Science Applications International Corp.
Monterey, CA 93940

20. Professor Neil Rowe, Code 52Rp — 1
Department of Computer Science
Naval Postgraduate School
Monterey CA 93943-5000

Thesis
H11041 Hagaman
c.1 A prototype expert system to forecast typhoon conditions at Cubi Point, Philippines.